SPECTRAL ANALYSIS OF VLF/ELF SIGNALS

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

AMIT PRAKASH AGRAWAL

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

AUGUST, 1982

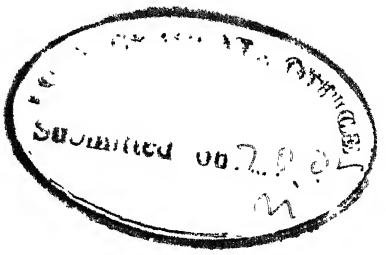
DEDICATED

TO

MY DEAREST

ELDERS

EE- 1982- M-AGA- SPE



CERTIFICATE

Certified that the work entitled 'SPECTRAL ANALYSIS OF VLF/ELF SIGNALS' by Mr. Amit Prakash Agrawal has been carried out under my supervision and has not been submitted elsewhere for the award of a degree.

R. Raghuram
Asstt. Professor
Department of Electrical Engineering
Indian Institute of Technology
Kanpur

ACKNOWLEDGEMENTS

I express my deep sense of gratitude and appreciation to Dr. R. Raghuram for his never flagging enthusiasm, encouragement, constructive criticism and invaluable guidance provided by him throughout the course of my thesis.

I am thankful of my friend Mr. Govind for his valuable discussions which I had with him in the hardware part of the work.

Finally I thank Mr. K.K. Misra for his efficient typing work.

AMIT P AGRAWAL

CONTENTS

ABSTRACT

The objective of this project is to analyse VLF/ELF or speech signals in real time for its various frequency components using digital signal processing techniques. Fourier transform is used as a useful tool in extracting information in speech signals. Analog methods have limite resolution and flexibility. Disadvantages are overcome by taking Discrete Fourier Transform (DFT) FFT is an efficie way of realizing DFT.

The FFT is computed through the system TM 990/189 microcomputer. It consists of TMS 9980 CPU and TMS 9901 PSI (Programmable System Interface). The analog signal of low frequency is converted into digital signal which is stored in memory of the system through PSI. The output points of FFT stored in memory, are converted into analog signal and displayed on the oscilloscope. Thus the spectrum of analog signal in VLF/ELF range can be analysed.

CHAPTER 1

INTRODUCTION

Analysis of signals in the frequency domain is an important measurement concept which is widely used for providing electrical and physical system performance informations. There are many instances when signal processing involves the measurement of spectra. For example, in speech recognition problems, spectrum analysis is usually a preliminary to further acousting processing. In speech bandwidth reduction systems, spectral analysis is generally the basic measurement. In sonar systems, sophisticated spectrum analysis is required for the location of surface vessels and submarines. In radar systems, obtaining target velocity information generally implies the measurement of spectra. Thus spectrum analysis encompasses a great variety of different measurements. Several types of instruments are used for frequency response signal analysis e.g. spectrum analyser, digital Fourier analysers, wave analysers, distortion analysers and modulation analysers.

Each of the instruments measures basic properties of a signal in the frequency domain, but each uses a different technique. The Fourier analyser uses digital sampling and transformation techniques to form a Fourier spectrum display that has amplitude as well as phase information. Our project also aims at analysing the spectrum of speech or VLF/ELF signals for real time applications.

In spite of the advantages of digital signal analysis, analog methods are used for spectrum analysis at higher frequencies. The reason for this widespread use of analog methods for making this class of measurements is the ease of construction of narrow band analog filters for reasonably high frequencies. To carry out the same operation digitally the input signal must first be sampled, and then each sample is converted to its digital equivalent. If the input signal is sampled at uniformly spaced instances, the sampling rate must be at least twice the highest frequency present in the analog input signal in order to avoid errors in the measurement. The high sampling rate required by this method makes the cost of the sample and hold and ADCs which must be used in the implementation of such methods prohibitive. These requirements make the spectrum analysis increasingly difficult at higher frequencies. Our project is the spectrum analysis of speech or VLF/ELF signals.

The algorithm used for the spectrum analysis is the Fast Fourier Transform (FFT). We have selected radix-2, in-place algorithm. The processer to compute FFT can be designed using multipliers; bit-slice microprocessors or ordinary microprocessors. The design of processor with multipliers involves more hardware. In the design of processor with bit-slice microprocessors, it is a difficult to write microprogram. Therefore we chose the ordinary

microprocessors. The 16-bit TMS 9980A microprocessor was selected for our project.

To take the input samples, a data acquisition system is designed. This system samples input analog signal and converts them into digital form and are stored in the system memory. The microcomputer does the FFT and output points are displayed on the oscilloscopes. The hardware system for display is designed. The number of points and scans are displayed on the Y-axis and X-axis respectively of the oscilloscope. The magnitude of FFT points are sent to the Z-axis of oscilloscope. Thus the magnitude of spectrum can be seen in the form of intensity on the screen.

The aim of the present work is to present a compact visual display of the spectral characteristics of signals in the VLF/ELF range. Such a display enables one to readily pick out features of the signal. An application is the measurement of magnetospheric electron densities using whistlers. Whistlers are the result of the dispersion resulting from the propagation of lightening strokes or atmospherics through the magnetosphere. A spectrogram of a whistler is shown in Fig 1.1. The minimum propagation delays give the electron densities along various paths while the corresponding frequencies give the paths themselves.

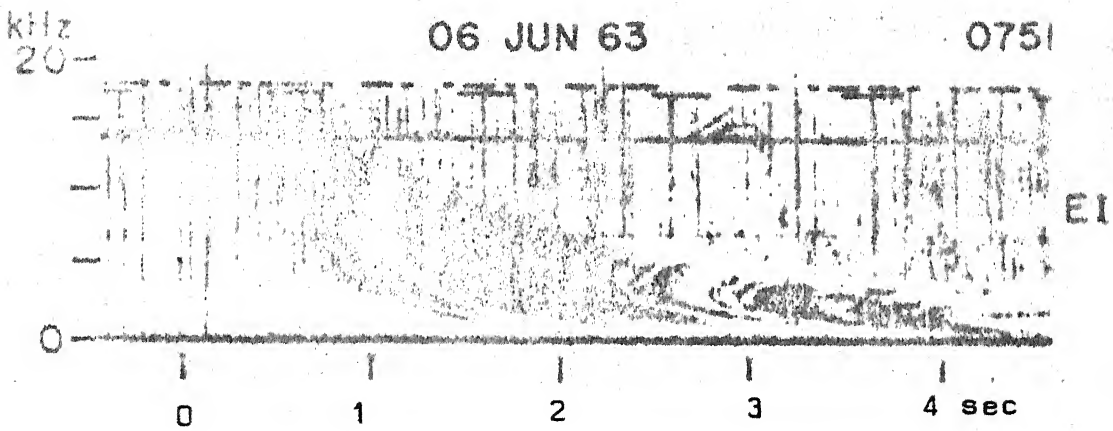


Fig 1.1 Spectrogram showing multipath whistlers recorded at eights station. The spectrogram shows several traces or components all of which originated from an atmospheric at 0 sec.

CHAPTER 2

FAST FOURIER TRANSFORM

FFT is an algorithm (i.e. a particular method of performing a series of computations) that can compute the discrete Fourier transform much more rapidly than other available algorithms. Simple calculations show that computing DFT of N complex points requires N^2 complex multiplications. The FFT owes its success to the fact that the algorithm reduces the number of multiplications and additions required in the computation of DFT.

An original algorithm developed by J.W. Cooley and J.W. Tukey reduces the computational load to $N \log_B N$ (complex multiplications), where B is the base (the base may be 2, 4, 8 or 16).

There have been some variations (like Sande-Tukey algorithm) to the original algorithm and at present there are many algorithms available for calculating the FFT. There have been many additional modifications depending on the particular requirements, the limitations of the available hardware, and the ingenuity of the individual designer or programmer.

Most of these algorithms may be classified as either (a) in place or (b) natural input-output [9]

2.1 IN PLACE ALGORITHM

An in-place algorithm is one in which a given component of any intermediate vector may be stored in the same location occupied by the corresponding component of the preceding vector. This type of algorithm requires less total storage, but at the same time the computational time is higher than that required for natural input-output algorithm. In this, either the output spectrum appears in an unnatural order or they require that the input data be arranged before entering the computation array.

Thus in-place algorithm requires the reordering of input or output data. The process in which the input points are reordered from natural to unnatural form is called scrambling. On the other hand the process in which the output points are reordered from unnatural to natural form is called unscrambling. In our system we need unscrarbling operation of output points.

2.2 SIGNAL FLOW GRAPH

FFT signal flow graph for in-place algorithm for N=8 is shown in Fig 2.1. It consists of data array and computational arrays. Data array is represented by a vertical column of nodes on the left of the graph. The vertical columns to the right of the data array correspond to computational arrays and in general there will be r computational arrays where $r = \mathrm{Log}_2 N$

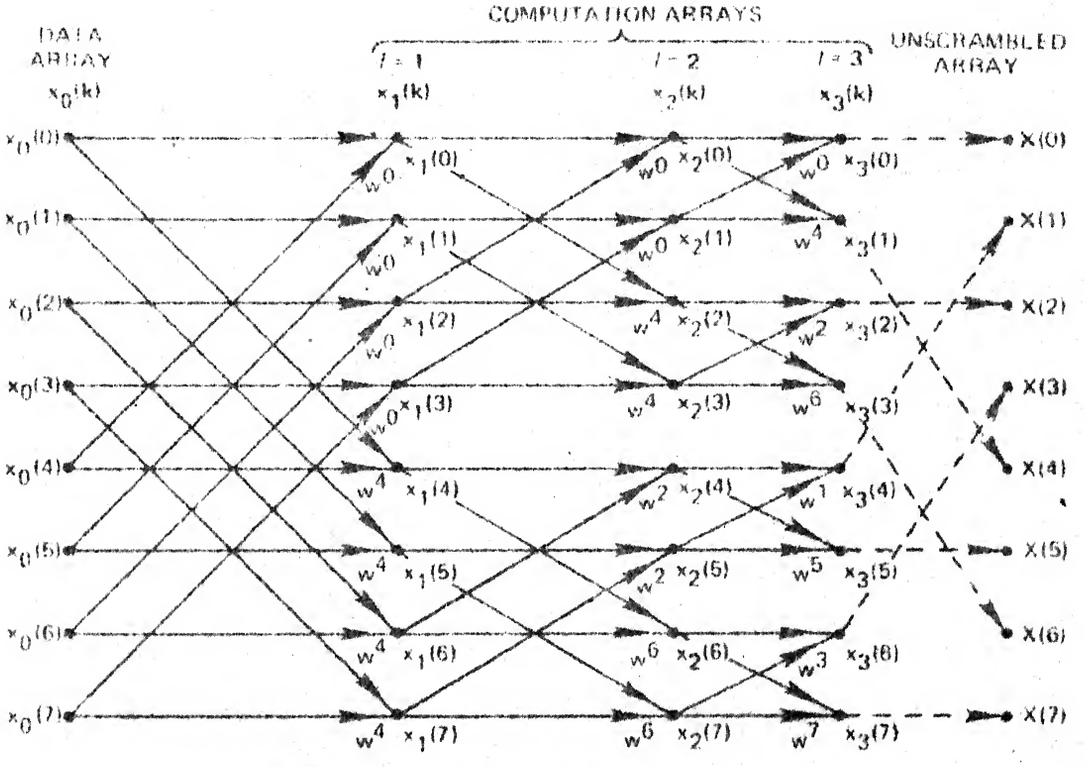


Figure 2·1 : FFT signal flow graph for $N = 8$.

2.3 BASIC PROPERTIES OF IN-PLACE ALGORITHM [1]

From the flow graph, the following basic properties can be identified :

(i) Number of computational arrays

$$r = \text{Log}_2 N$$

(ii) Each array requires $\frac{N}{2}$ complex multiplications and and N complex additions. Hence the total number of multiplications and additions required are $(N/2)\ \text{Log}_2 N$ and $N\ \text{Log}_2 N$ respectively. The ratio of direct to FFT computation time is

$$\frac{N^2}{N/2\ \text{Log}_2 N} = \frac{2N}{\text{Log}_2 N}$$

(iii) The computation of a dual node pair requires only one multiplication and two additions. If the weighting factor at one of the nodes in a dual node pair is W^P, then the weighting factor at the other node of the pair is $W^{P+N/2}$

$$W^{P+N/2} = -W^P$$

then

$$x_\ell(k) = x_{\ell-1}(k) + W^P\, x_{\ell-1}(k + N/2)$$

$$x_\ell(k + N/2) = x_{\ell-1}(k) - W^P\, x_{\ell-1}(k + N/2)$$

here $x_\ell(k)$ indicates k^{th} component in the ℓ^{th} array.

(iv) The spacing between dual node pairs differ from array to array. In the ℓ^{th} array (ℓ = 1, 2, ..r), the spacing is N/2 , i.e. x(k) and x(k+N/2) constitute a dual node pair.

(v) To evaluate the value of P, the exponent of W, for any index in a given array, the following procedure is followed. Represent k, the node index in the ℓ^{th} array, in binary form with r bits, retain the most significant bits and add (r-ℓ) leading zeros to form a r bit binary number. Reverse the bit order of the resulting number. The decimal equivalent of the final binary number gives the index P. The weighting factor for the k^{th} node of the ℓ^{th} array is W^P.

(vi) The output after r arrays is in scrambled form. To unscramble the output x(k), write the index k in binary form with r bits and reverse the bit order. The resulting decimal number is the index n of x(k). The unscrambling procedure is shown in Fig 2.2.

2.4 NATURAL INPUT-OUTPUT ALGORITHM

Natural input-output algorithm is one in which a given component of any intermediate vector may not be stored in the same location occupied by the corresponding component of the preceeding vector, thus requiring the extra memory for storing the intermediate result. An N point FFT

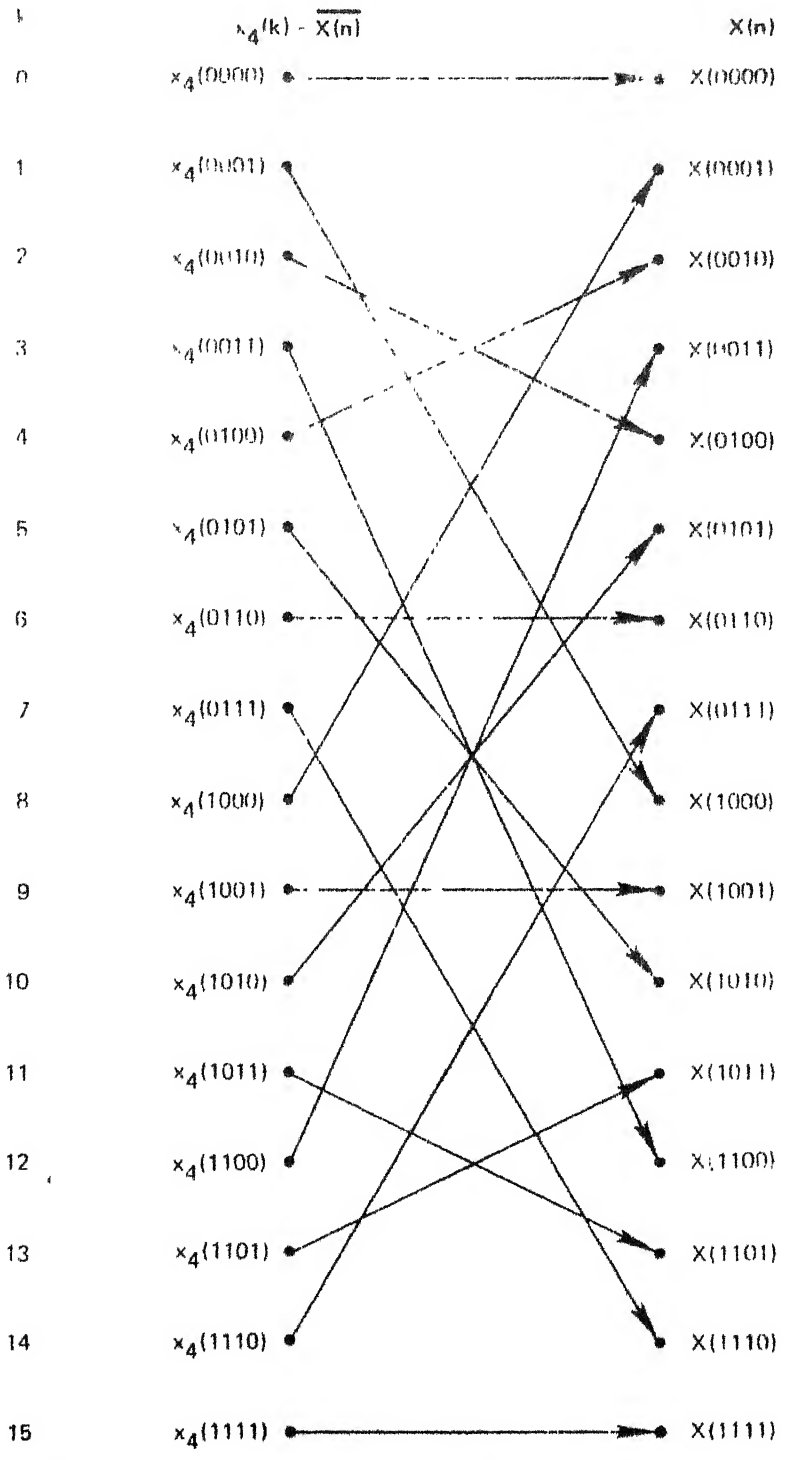


Figure 2-2 : Example of bit-reversing operation for $N = 16$.

(N real and N imaginary) will require 4N words of memory as compared to 2N words required for in-place algorithm. These algorithms, of course, maintain the natural input-output order and thus do not require scrambling/unscrambling at the input/output levels, and as such are faster as compared to the in-place algorithm. In our system, in-place algorithm has been used for it requires less number of memory locations.

2.5 DECIMATION-IN-TIME/FREQUENCY [7]

The decimation-in-time FFT algorithms are all based upon the decomposition of the DFT computation by forming smaller and smaller subsequences of the input sequence x(k). Alternatively we can consider dividing the output sequence, x(n), into smaller and smaller subsequences in the same manner. The class of FFT algorithms based on this procedure is commonly referred to as decimation in frequency.

2.6 FFT ALGORITHMS FOR REAL DATA [1]

In applying FFT, we often consider real function of time whereas the frequency functions are, in general complex. Thus a single computer program can be written to determine both the discrete transform and its inverse such that a complex time waveform is assumed.

If the time function being considered is real, we must set to zero the imaginary part of the complex time function.

This approach is inefficient in that the computer program will still perform the multiplications for the zero imaginary parts. In following sections two techniques for using this imaginary part of the complex time function to more efficiently compute the FFT of real functions are described.

2.7 FFT OF TWO REAL FUNCTIONS SIMULTANEOUSLY [1]

For efficient computer program, it is desired to compute the DFT of the real time functions h(k) and g(k) from the complex function

$$y(k) = h(k) + jg(k)$$

That is, y(k) is constructed to be the sum of two real functions where one of these real functions is taken to be imaginary. After computing DFT of y(k), the real and imaginary parts of the DFT are decomposed. Thus the simultaneous discrete transform of two real time functions can be accomplished.

2.8 TRANSFORM OF 2N SAMPLES WITH N SAMPLE TRANSFORM [1]

The imaginary part of the complex time function can also be used to compute more efficiently the DFT of a single real time function. Consider a function x(k) which is described by 2N samples. We wish to break the 2N point function x(k) into two N sample functions. We devide x(k) as follows :

$$h(k) = x(2k)$$

$$g(k) = x(2k+1) \qquad k = 0,1, \ldots\ldots N-1$$

$$y(k) = h(k) + jg(k)$$

That is, function h(k) is equal to the even numbered samples of x(k), and g(k) is equal to the odd numbered samples. The computation procedure is given in Fig 2.3.

A general FFT flowchart for N complex input points based on the algorithm of section 2.1 is shown in Fig 5.2. The flowchart has been implemented for our system. Since the input points are real, imaginary parts are made zero at input stage. The algorithm described in section 2.8 is more efficient for the imaginary parts of the complex time function can also be used to compute DFT of a real time function.

1. Function x(k) is real $\quad k = 0,1, \ldots .2N-1$

2. Divide x(k) into two functions

$$h(k) = x(2k) \qquad k = 0,1, \ldots .N-1$$

$$g(k) = x(2k+1)$$

3. Form the complex function

$$y(k) = h(k) + jg(k) \quad k = 0,1, \ldots .N-1$$

4. Compute

$$Y(n) = \sum_{k=0}^{N-1} y(k)\, e^{-j2\ell rK/N}$$

$$= R(n) + jI(n) \quad n = 0,1, \ldots N-1$$

where R(n) and I(n) are the real and imaginary parts of Y(n), respectively.

5. Compute

$$Xr(n) = \left[\frac{R(n)}{2} + \frac{R(N-n)}{2}\right] + \cos\frac{\pi n}{N}\left[\frac{I(n)}{2} + \frac{I(N-n)}{2}\right]$$

$$- \sin\frac{\pi n}{N}\left|\frac{R(n)}{2} - \frac{R(N-n)}{2}\right|, n=0,1, \ldots .N-1$$

$$Xi(n) = \left[\frac{I(n)}{2} - \frac{I(N-n)}{2}\right] - \sin\frac{\pi n}{N}\left[\frac{I(n)}{2} + \frac{I(N-n)}{2}\right]$$

$$- \cos\frac{\pi n}{N}\left[\frac{R(n)}{2} - \frac{R(N-n)}{2}\right], n=0,1, \ldots .N-1$$

where Xr(n) and Xi(n) are real and imaginary parts of 2N point of x(k).

Fig 2.3: Computation procedure for DFT of a 2N point function by means of an N point transform.

CHAPTER 3

SYSTEM DESIGN CONSIDERATIONS

3.1 CONSIDERATIONS IN FFT HARDWARE IMPLEMENTATION [2]

One needs to examine the following four factors :

(a) The reasons for building special-purpose-hardware.

(b) What are the various options available, so far as machine organization is concerned.

(c) Trade-offs between cost, speed and accuracy.

(d) Interface units etc.

3.2 SYSTEM BLOCKS

The block diagram of the system is shown in Fig 3.1. The system consists of three main blocks: (1) data aquisition system, (2) FFT processor and (3) display system.

3.3 FFT PROCESSOR [7]

The organization of an FFT processor is usually dictated by the performance and cost requirements and the technology assumed. An N-point FFT requires $\frac{N}{2} \log_2 N$ basic computations where one basic computation implies one complex multiplication, one complex addition and one complex subraction. Depending upon the number of basic computations done in parallel, four families of machine organizations are commonly used.

(a) SEQUENTIAL PROCESSOR:

The main features of the sequential processor are :

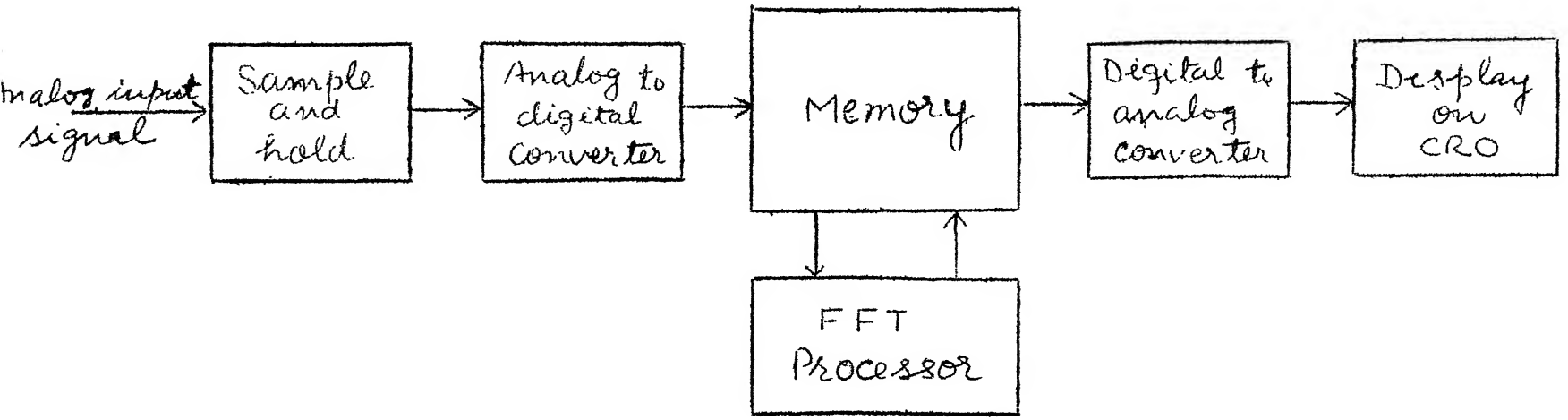


FIG 3.1 : SYSTEM BLOCK DIAGRAM

(i) One Arithmatic unit

(ii) $\frac{N}{2} \cdot \log_2 N$ operations sequentially.

(iii) Maximum sampling rate for real time processing $= \frac{1}{2(N-1)t}$ MHz, t = μsec, time required by a basic computation.

(iv) Execution time = $\frac{N}{2} \cdot \log_2 N . t$ μsec.

(b) CASCADE PROCESSOR

The sequential processor does not yield speeds generally required for real time processing of signal. Its performance can be improved by introducing a certain amount of parallelism. The main features of a cascade processor are:

(i) $\text{Log}_2 N$ arithmatic units for $\log_2 N$ operations in parallel.

(ii) $\frac{N}{2}$ operations sequentially.

(iii) Maximum sampling rate allowable = $\frac{1}{t}$ MHz, t = μsec.

(c) THE PARALLEL ITERATIVE PROCESSOR

A second alternative for improving the performance of a sequential processor is to introduce parallelism within each iteration i.e. performing all the computations within an iteration in

parallel and $\log_2 N$ iterations sequentially. Such an organization is known as parallel iterative. The main features of this organizations are :

(i) $\frac{N}{2}$ Arithmatic units for $\frac{N}{2}$ operations in parallel.

(ii) $\log_2 N$ operations sequentially.

(iii) Max. sampling rate allowable = $\frac{N}{(\log_2 N).t}$ MHz

(iv) Execution time = $(\log_2 N).t$ μsec.

(d) THE ARRAY ANALYSER

The organization in which $\frac{N}{2}.\log_2 N$ basic computations are done in parallel is known as an array analyser. Due to high degree of parallelism, it yields a very high through-put rates. However, the cost of this approach limits its application. Its main features are :

(i) $\frac{N}{2}.\log_2 N$ a.u. for $\frac{N}{2}.\log_2 N$ operations in parallel.

(ii) Max. sampling rate allowable = $\frac{N}{t}$ MHz

(iii) Execution time = t μsec.

Reviewing the processors as mentioned above the sequential processor has been selected. It is slowest but cheapest. This processor can be designed with microprocessor. A microprocessor has been used because of flexibility and cost.

3.4 SYSTEM DESIGN CONSTRAINTS [8]

a) REAL TIME REQUIREMENT

Suppose we want to go upto VLF/ELF or speech signals of 5KHz, the minimum sampling rate will be 10 KHz (minimum sampling rate is double of the input signal frequency). If a 1024 point transform is chosen, a time of 0.1 sec will be available for input, processing and output for real time computation.

b) MEMORY REQUIREMENTS

(i) Program memory

(ii) SINE, COS values for computation of weight

(iii) Values for windowing = N

(iv) Data memory = N or 2N depending on algorithm. Here N is number of sample points.

c) COMPUTATION REQUIREMENTS

(i) $\frac{N}{2} \log_2 N$ complex multiplications or $2N \log_2 N$ real multiplications

(ii) $N \log_2 N$ complex additions or subtraction or $2N \log_2 N$ real additions or subtraction.

(iii) Bit reversal required for calculation of weights and for scrambling input/output depending upon the algorithm used.

d) ACCURACY:

Minimum of 16-bit processing accuracy is required. The detail is given in section 3.7.

3.5. CHOICE OF MICROPROCESSOR

An 8x8 bit multiplication sub-routine on 8080 processor takes about 200 micro-seconds. Since we are doing 16 bit processing, we will need to execute 8x8 bit sub-routine four times. So time per 16x16 bit multiplication will be 800 micro-seconds. For $\frac{N}{2} \log_2 N$ complex multiplications or $2N \log_2 N$ real multiplications, total time required will be

$$4 \times 200 \times 2 \times 1024 \times 10 = 15384 \text{ msec.}$$

Thus it is necessary to use an external fast multiplier if we want to meet the real time requirement of 0.1 sec.

In order to do one 16x16 bit multiplication on 8080 using external multiplier, 8 MOV instructions and 8 load instructions are required. Thus it will take 4000 m sec. to do $2N \log_2 N$ real multiplications. These are optimistic estimates.

The basic reason why 8080 is slow, is that we require double precision (16 bit) accuracy and 8080 is only 8 bit processor. The same problem will be with other 8 bit processors like 8085 etc. Hence it is necessary to go for a 16 bit processor. TMS 9980 microprocessor which is a 16 bit processor, has been chosen for the project. This processor has in-built multiplication routine (MPY). It does 16x16 bit unsigned multiplication.

The multiplier-interfacing to the system (TM 990/189) is not advantageous because the time taken for one multiplication by interfaced multiplier is more than that of in-built multiplication routine. One MPY instruction takes 62 cycles. The execution time for one MPY instruction can be computed [5] as follows :

$$T = t_c(\emptyset)\ (C+W \cdot M)$$

where

T = total instruction time

tc(Ø) = clock cycle time

C = number of clock cycles for instruction execution plus address modification

W = number of required wait states per memory access for instruction execution plus address modification.

M = number of memory accesses

Here

$t_c(\emptyset)$ = 0.400 micro-seconds

C = 62

M = 10

W = 0

Hence T = 0.400 (62 + 0.10)

= 24.8 micro-seconds

If we interface the fast multiplier chip with the system (TM 990/189) through PSI, one LDCR and one STCR instructions

along with other instructions will be used for one multiplication. We consider only LDCR and STCR instructions. The execution time for these instructions can be computed as follows :

For LDCR; C = 58, M = 6, W = 0

$$T_L = 0.400\ (58 + 0.6)$$

$$= 23.2 \text{ micro-seconds}$$

For STCR, C = 68, M = 8, W = 0

$$T_S = 0.400\ (68 + 0.6)$$

$$= 27.2 \text{ micro-seconds.}$$

$$T = T_L + T_S = 23.2 + 27.2 = 50.4 \text{ micro-seconds}$$

Hence it can be concluded that the interfaced multiplier takes double time than that of in-built multiplier. Therefore we have used in-built multiplication routine.

3.6 DATA ACQUISITION AND DISPLAY SYSTEMS

The schemes for both the data acquisition and the display systems are shown in Fig 3.2. These schemes have been implemented and tested.

The PSI plays the key role for operating both the systems. It generates the control signals via software of the microprocessor system. The pins P8(LSB) through P15(MSB) of the PSI have been used as input output lines.

The input analog signal after sampling is digitized by analog to digital converter (ADC). The start convert pulse to ADC is generated by the monostable circuit. The trigger input to monostable is generated by pin P6 of the PSI. The EOC pulse of ADC controls the switch of the sample and hold. The outputs of ADC are latched and are read and stored in the system memory by the microprocessor. During this operation the pin P5 is high and enables the latch L1 and the buffer B2. So the display system remains off.

The microprocessor performs the FFT computations of the stored input points. The output points are again stored in the same memory locations where input sample points were stored. To display these points the pin P5 becomes low which enables the latch L2 and the buffer B3. So the data acquisition system remains off. Now the clocks at P6 goes to the counter C1. The output points appear on

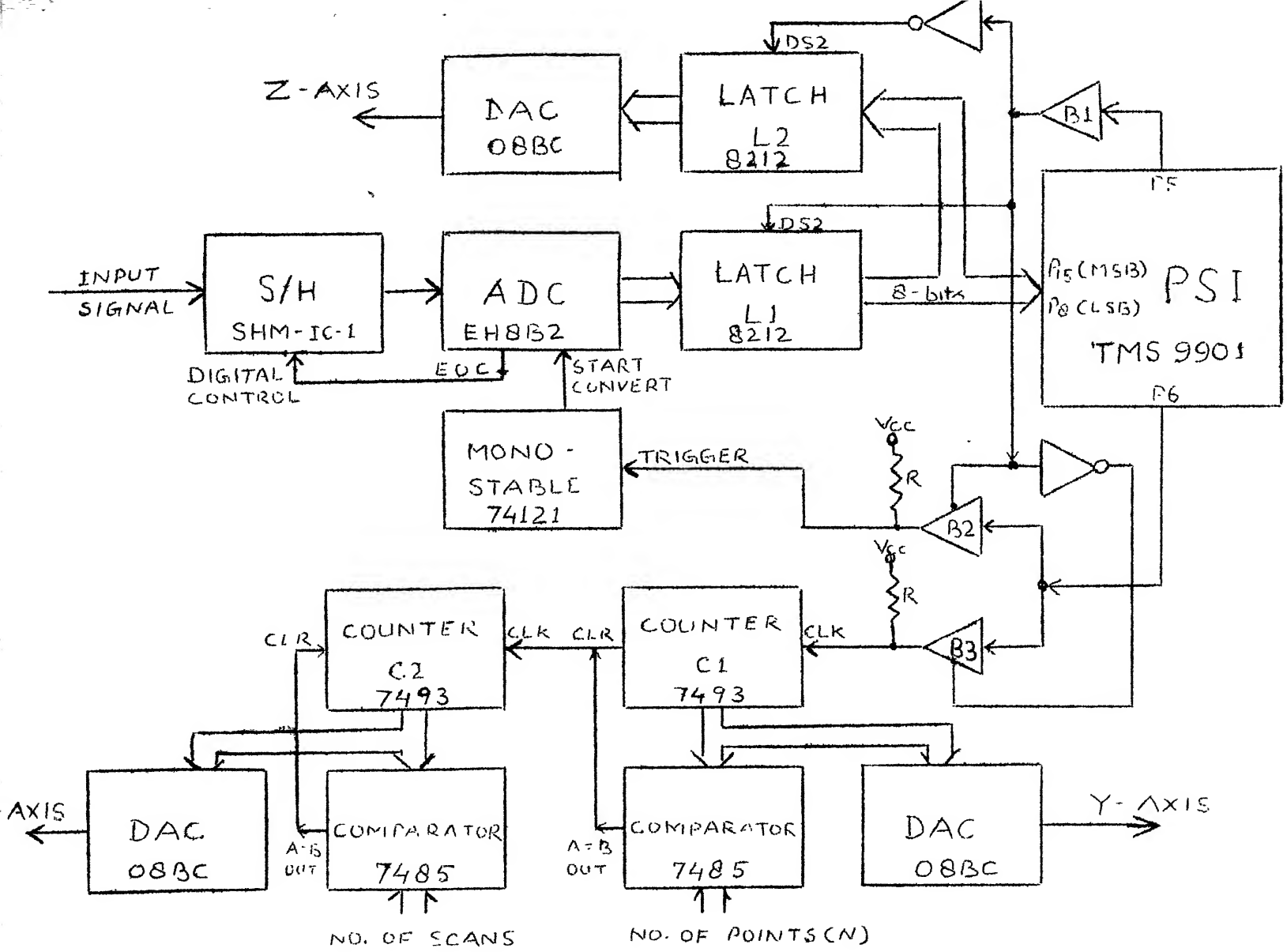


FIG 3.2 : SCHEME FOR DATA ACQUISITION AND DISPLAY SYSTEMS

the I/O pins of the PSI one by one at each clock. The 8-bit output of each FFT point is converted into analog form by digital to analog converter (DAC) and it goes to the Z-axis of the oscilloscope. At the same time the output of the counter C1 through DAC is displayed on the Y-axis of the oscilloscope. Since there is no clock to the counter C2, the output that goes to the X-axis of the oscilloscope is zero. Since the FFT computation is only for N=32, one clock appears at the counter C2 after the display of one set of 32 FFT points. This clock will shift the beam point right on the X-axis and the same operation is repeated and the second set of FFT points are displayed on the screen and so on. The intensity on the Y-axis displays the magnitude of the FFT points.

The timing diagrams for the data acquisition and the display systems are shown in Fig 3.3(a) and 3.3(b). The chapter four and five describe respectively the hardware and software part of the data acquisition and the display systems.

3.7 EIGHT BITS VERSUS SIXTEEN BITS ACCURACY

With 8 bits in fractional part the precision of 8 bits is given by

$$\text{Precision (8 bits)} = 8 \times \log_{10}2 = 2.41 \text{ digits}$$

By increasing the number of bits in the fractional part a more accurate representation for the spectrum can

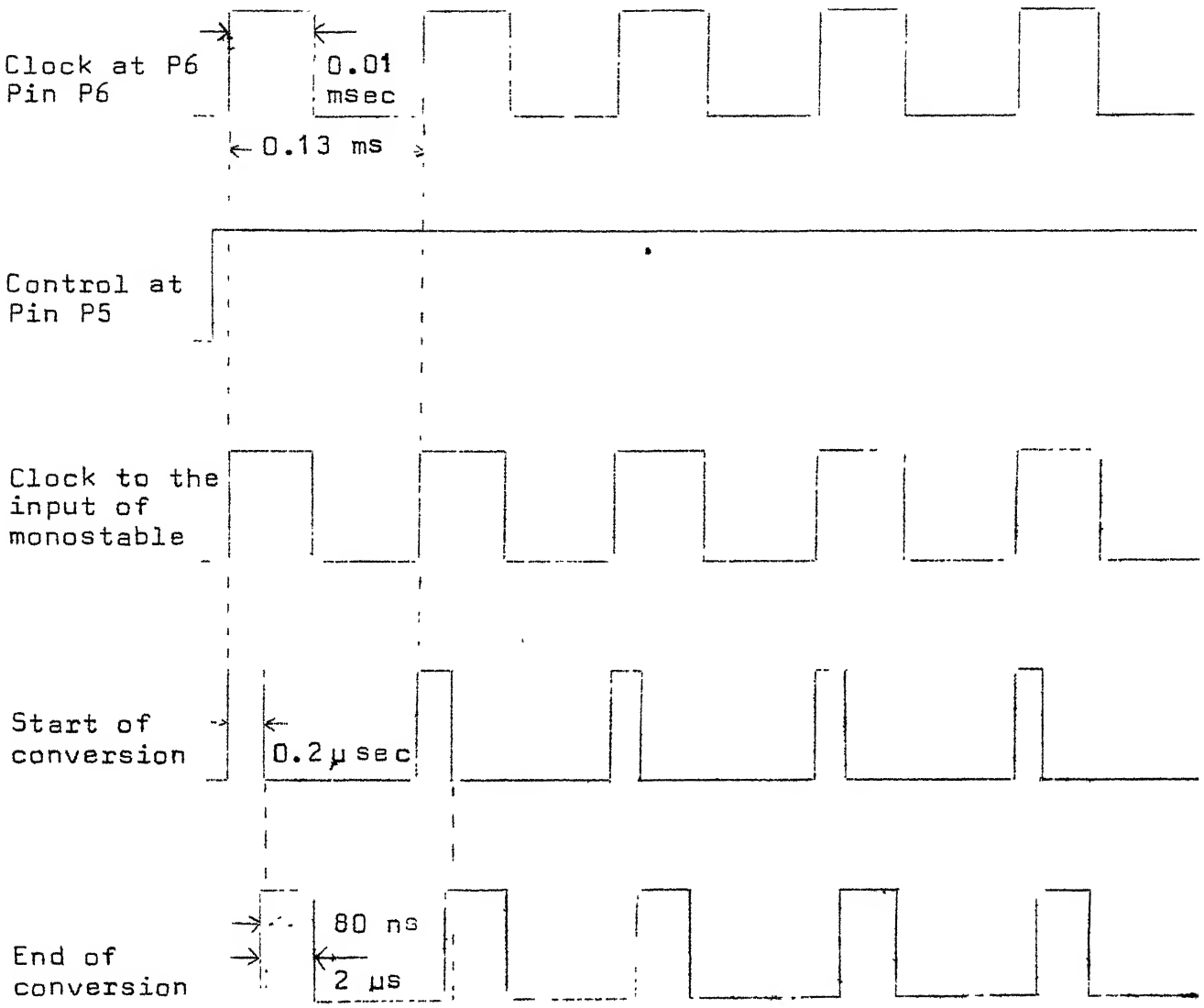


Fig 3.3(a) Timing diagram of the Data Acquisition System

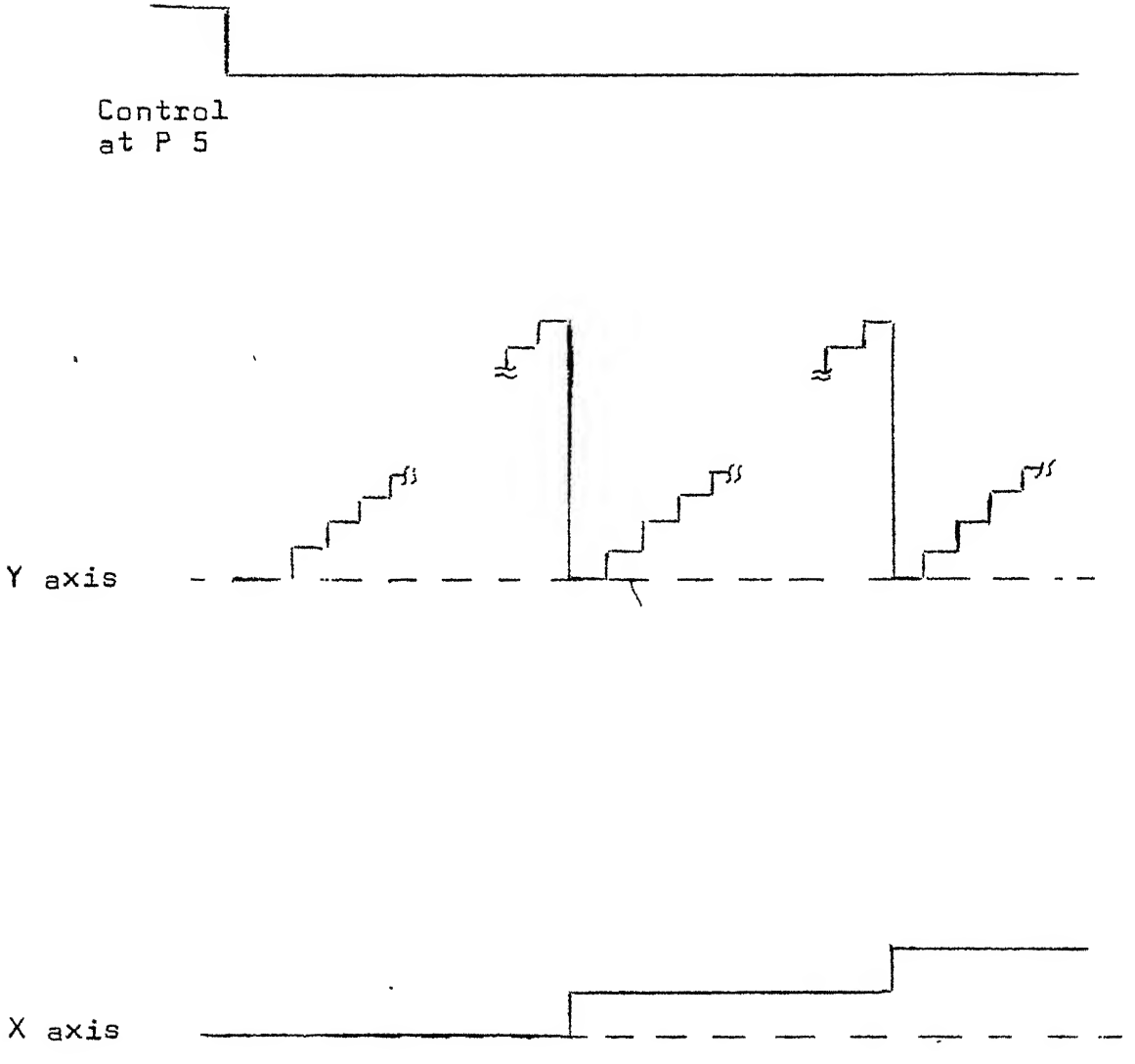


Fig. 3.3(b) Timing diagram for display system

be obtained. Also computation time considerations dictate that the maximum number of bits be limited. With 16 bits, sufficient accuracy of representation can be obtained without degrading the computation time to a large extent.

With 16 bits, precision of 16 bits can be calculated as

$$\text{Precision (16 bits)} = 16 \times \log_{10}2 = 4.82 \text{ digits}$$

This is quite sufficient for the envisaged use. Therefore 16 bits accuracy for the computation of FFT has been chosen for our project.

CHAPTER 4

DESCRIPTION OF THE SYSTEM

4.1 MICROPROCESSOR KIT

The system used is Texas Instruments TM 990/189 [5]. It is a self-contained, single-board microcomputer system. The system features include:

- TMS 9980A microprocessor.
- 1024 bytes of RAM expandable on board to 2048
- Bytes (each byte contains 8 bits of data)
- 4096 bytes of ROM expandable on board to 6144 bytes
- 2 MHz crystal controlled clock
- Audio cassette interface
- 16 bit programmable I/O port and interrupt monitor (TMS 9901)
- 45 - key alphanumeric keyboard
- Ten-digit, seven-segment L.E.D. type alphanumeric display
- Visual and acoustic indicators
- Resident system monitor and assembler
- Single step instruction execution

In addition to onboard memory expansion, two other system expansion options are available :

- A TMS 9902 asynchronous communications controller, and accompanying interface circuits for either RS-232-C or 20 mA current loop terminals can be added.

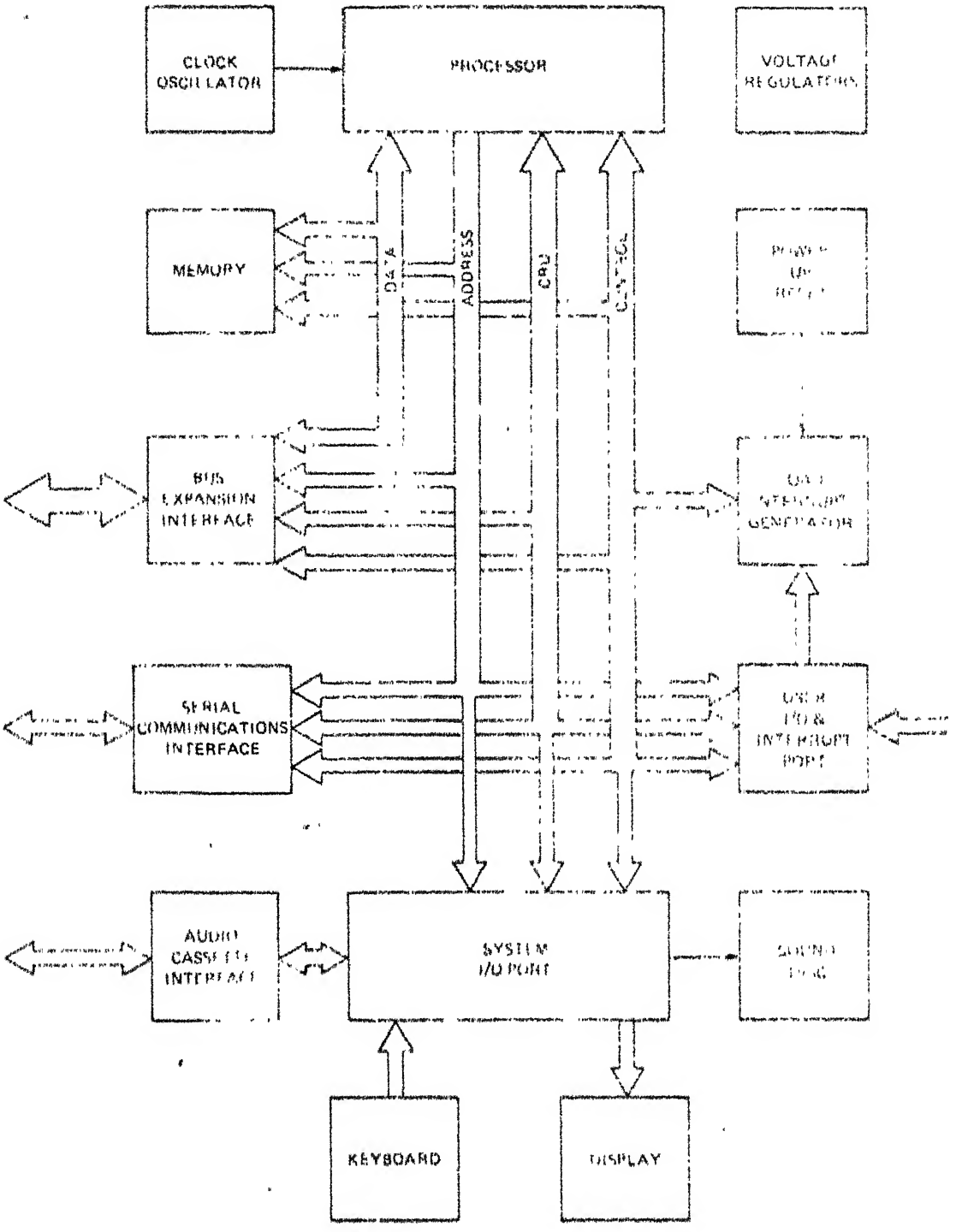


Figure 4-1 : System Block Diagram

- The bus can be expanded by use of the Bus Interface.

Figure 4.1 shows the system architecture along with the user options.

4.1.1. C.P.U.

The TMS 9980A is a software-compatible member of TI's 9900 family of microprocessors. The key features are:

- 16 Bit Instruction word
- Full Minicomputer Instruction Set Capability Including Multiply and Divide (multiplication and division)
- Upto 16,384 Bytes of Memory
- 8 Bit memory data bus
- Advanced Memory to Memory Architecture
- Separate Memory, I/O, and Interrupt. Bus Structures
- 16 General Registers
- 4 Prioritized Interrupts
- Programmed and DMA I/O capability
- On-chip 4 phase clock Generator
- 40 pin package
- N-Channel Silicon-Gate Technology

A word is defined as 16 bits or 2 consecutive bytes in memory. The words are restricted to be on even address boundaries, i.e. the most significant half (8 bits) resides at even address and the least significant half at the subsequent odd address. A byte can reside at even or odd address.

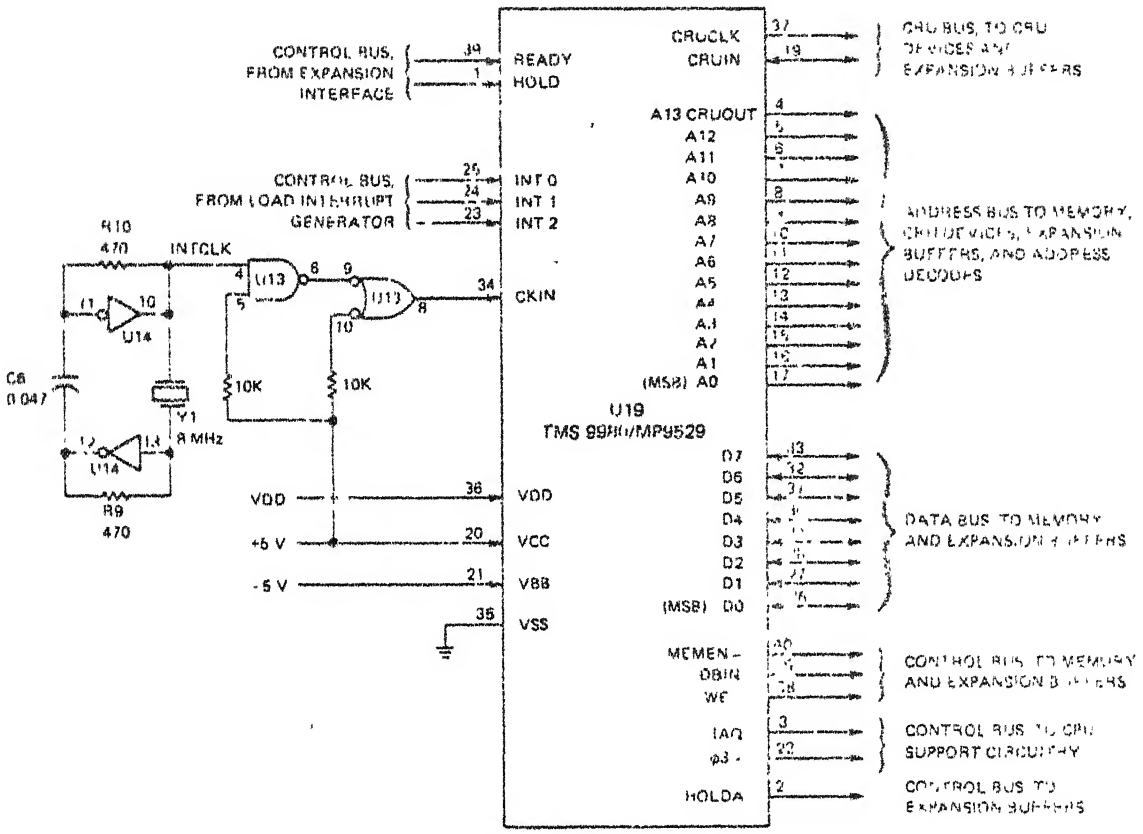


Figure 4-2 TMS 9980A Signals

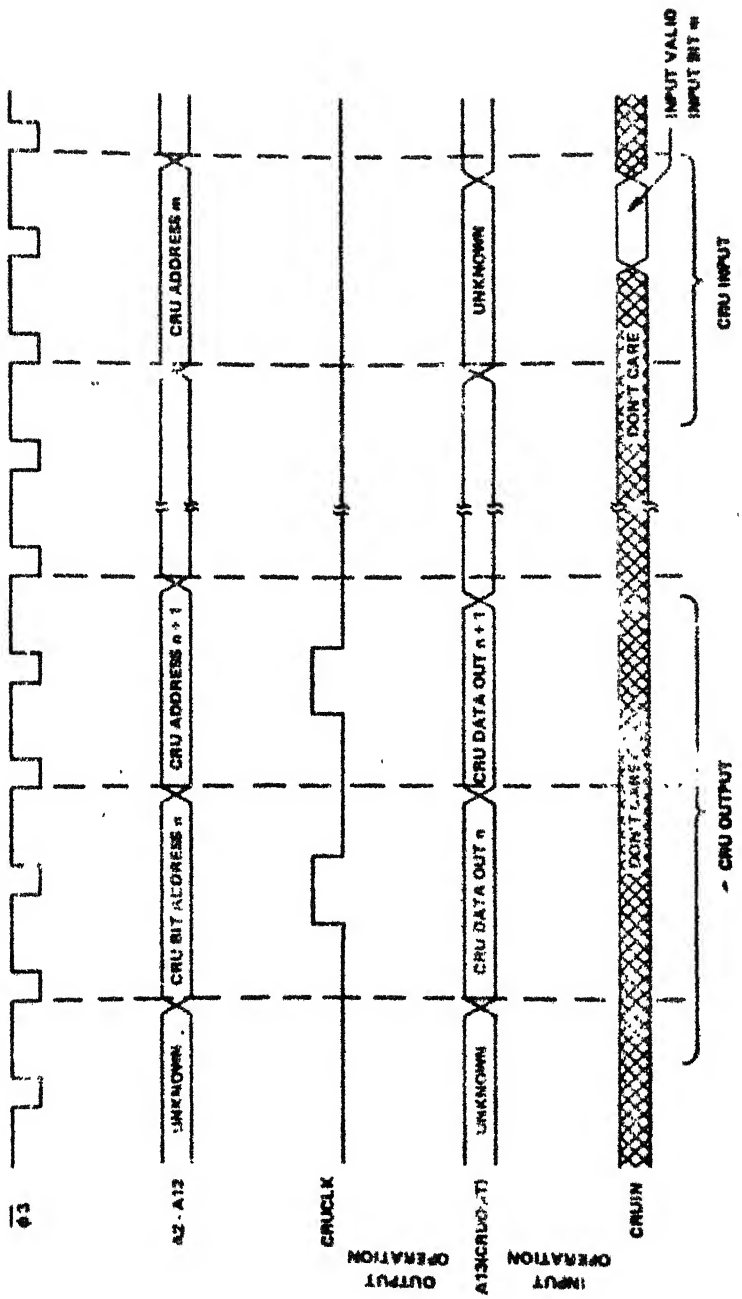


FIGURE 4-3- TMS 9900A/TMS 9981 CRU INTERFACE TIMING

The microprocessor signals are shown in Fig 4.2. The TMS 9980A CRU Interface Timing is shown in Fig 4.3.

4.1.2 PROGRAMMABLE SYSTEM INTERFACE (PSI)

It is a multifunctional component designed to provide low cost interrupt and I/O ports and an interval timer for TMS 9900 - family microprocessor systems. The key features are :

- 9900 - family peripheral
- Performs interrupt and I/O Interface functions:
  - (i) Six dedicated interrupt lines
  - (ii) Seven dedicated I/O lines
  - (iii) Nine programmable lines as I/O or interrupt
  - (iv) Up to 15 interrupt lines
  - (v) Up to 22 input lines
  - (vi) Up to 16 output lines
- Easily cascaded for expansion
- Interval or Event Timer
- Single 5v power supply
- All inputs and outputs TTL-Compatible

The TMS 9901 Pin assignments and functions are shown in Fig 4.4. The TMS 9901 can be divided into three sub-systems: CRU interface, input/output interface and interrupt interface.

a) CRU INTERFACE

The CPU communicates with the TMS 9901 PSI via the CRU. The typical TMS 9901 application is shown in Fig 4.5. The

SIGNATURE	PIN	I/O	DESCRIPTION
$\overline{INTREQ}$	11	OUT	INTERRUPT Request When active (low) $\overline{INTREQ}$ indicates that an enabled interrupt has been received. $\overline{INTREQ}$ will stay active until all enabled interrupt inputs are removed.
IC0 (MSB)	15	OUT	Interrupt Code lines. IC0 IC3 output the
IC1	14	OUT	binary code corresponding to the highest
IC2	13	OUT	priority enabled interrupt. If no enabled
IC3 (LSB)	12	OUT	interrupts are active IC0—IC3 = (1,1,1,1).
$\overline{CE}$	5	IN	Chip Enable When active (low) data may be transferred through the CRU interface to the CPU. $\overline{CE}$ has no effect on the interrupt control section.
S0	39	IN	Address select lines The data bit being
S1	38	IN	accessed by the CRU interface is specified
S2	35	IN	by the 5-bit code appearing on S0-S4.
S3	25	IN	
S4	24	IN	
CRUIN	4	OUT	CRU data in (to CPU). Data specified by S0-S4 is transmitted to the CPU by CRUIN .When $\overline{CE}$ is not active CRUIN is in a high-impedance state.
CRUOUT	2	IN	CRU data out (from CPU) When $\overline{CE}$ is active, data present on the CRUOUT input will be sampled during CRUCLK and written into the command bit specified by S0-S4
CRUCLK	3	IN	CRU Clock (from CPU) CRUCLK specifies that valid data is present on the CRUOUT line.
$\overline{RST1}$	1	IN	Power Up Reset When active (low) $\overline{RST1}$ resets all interrupt masks to "0", resets IC0 — IC3 = (0, 0, 0, 0), $\overline{INTERQ}$ = 1, disables the clock, and programs all I/O ports to inputs $\overline{RST1}$ has a Schmitt-trigger input to allow implementation with an RC circuit as shown in Figure 7
V_{CC}	40		Supply Voltage +5 V nominal.
V_{SS}	16		Ground Reference
$\overline{\phi}$	10	IN	System clock ($\overline{\phi}3$ in TMS 9900 system, $\overline{CKOUT}$ in TMS 9980 system).
$\overline{INT1}$	17	IN	Group 1, interrupt inputs When active (Low) the signal is ANDed with its corresponding mask bit and if enabled sent to the interrupt control section. $\overline{INT1}$ has highest priority.
$\overline{INT2}$	18	IN	
$\overline{INT3}$	9	IN	
$\overline{INT4}$	8	IN	
$\overline{INT5}$	7	IN	
$\overline{INT6}$	6	IN	
$\overline{INT7}$/ P15	34	I/O	Group 2, programmable interrupt (active low) or I/O pins (true logic). Each pin is individually programmable as an interrupt, an input port, or an output port.
$\overline{INT8}$/ P14	33	I/O	
$\overline{INT9}$/ P13	32	I/O	
$\overline{INT10}$/P12	31	I/O	
$\overline{INT11}$/P11	30	I/O	
$\overline{INT12}$/P10	29	I/O	
$\overline{INT13}$/P9	28	I/O	
$\overline{INT14}$/P8	27	I/O	
$\overline{INT15}$/P7	23	I/O	
P0	38	I/O	Group 3, I/O ports (true logic): Each pin is individually programmable as an input port or an output port.
P1	37	I/O	
P2	26	I/O	
P3	22	I/O	
P4	21	I/O	
P5	20	I/O	
P6	19	I/O	

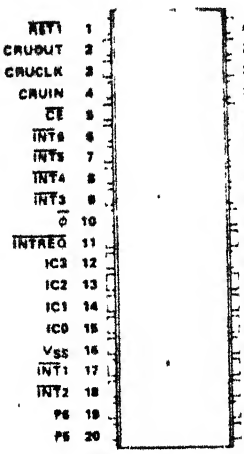


Fig 4.4 TMS 9901 Pin Assignments and Functions

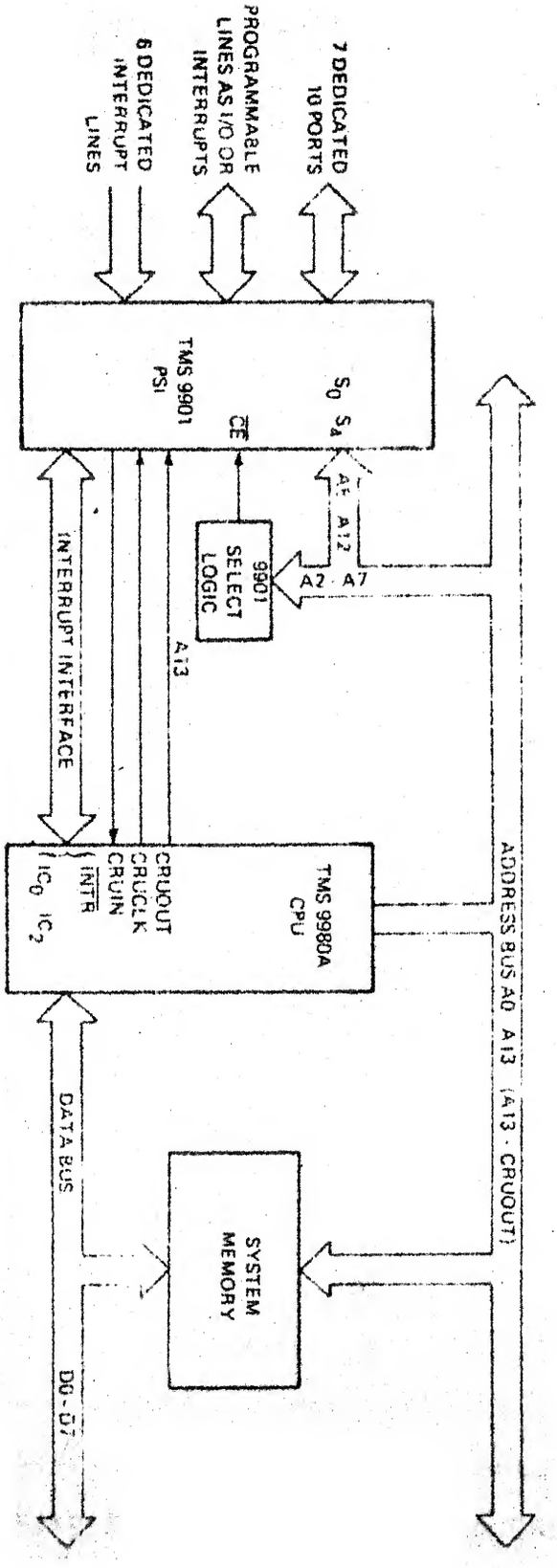


Figure 4-5: Typical TMS 9901 Programmable System Interface (PSI) Application

TMS 9901 occupies 32 bits in CRU read space and 32 bits in CRU write space. The Table 4.1 shows the mapping for CRU bit addresses to TMS 9901 function. The CRU interface consists of five address select lines (S0-S4), chip enable ($\overline{CE}$) and the three CRU lines (CRUIN, CRUOUT, CRUCLK). In the case of a write operation, the TMS 9901 strobes data off the CRUOUT line with CRUCLK. For a read operation, the data is sent to CPU on CRUIN line. The chip enable is generated by decoding the high-order address bits (A0-A9) on CRU cycles.

CRU is a bit addressable (4096 bits), synchronous, serial interface over which a single instruction can transfer between one and 16 bits serially. Each one of the 4096 bits of the CRU space has a unique address and can be read and written. During multi-bit CRU transfers, the CRU address is incremented at the beginning of each CRU cycle to point to the next consecutive CRU bit. When a 9980 CPU executes a CRU instruction, the processor uses the contents of workspace register 12 as a base address. The CRU address is brought out on the 15-bit address bus; this means that the least significant bit of R12 is not brought out of the CPU. During CRU cycles, the memory control lines ($\overline{MEMEN}$, $\overline{WE}$ and DBIN) are all inactive; $\overline{MENEN}$ being inactive (High) indicates the address is not a memory address and therefore is a CRU address or external instruction code. Also when $\overline{MEMEN}$ is inactive (High) and a valid address is present, address bits A0-A2 must all be zero to constitute a valid CRU address.

b) INPUT/OUTPUT INTERFACE

Upto 16 individually controlled, I/O lines are available and the unused dedicated interrupt lines also can be used as input lines. After reset all I/O lines are programmed as inputs. By writing to any I/O line, that line will be programmed as an output line until another reset occurs. An output line can be read and indicates the present state of the pin. A pin programmed to the output mode can not be used as an input pin. Applying an input current to an output pin may cause damage to the TMS 9901. The TMS 9901 outputs are latched and buffered off chip, and inputs are buffered onto the chip.

c) PROGRAMMABLE PORTS

A total of nine pins ($\overline{INT}7$/P15-$\overline{INT}15$/P7) on the TMS 9901 are user-programmable as either I/O lines or interrupts. Any pin which is not being used for interrupt should have the appropriate interrupt mask disabled (mask=0) to avoid erroneous interrupts to the CPU. To program one of the pins as an interrupt, its interrupt mask simply is enabled and the line may be used as if it were one of the dedicated interrupt lines.

d) SOFTWARE INTERFACE

Figure 4.6 lists the TMS 9980 code needed to control the TMS 9901 PSI. The code initializes the PSI to an eight-bit input port, and an eight bit output port.

ASSUMPTION:

- System uses clock at maximum interval (349 msec @ 3MHz)
- Interrupts 1-6 are used
- Eight bits are used as an output port, P0 – P7
- Eight bits are used as an input port, P8 – P15
- $\overline{\text{RST1}}$ (power-up reset) has been applied
- The most significant byte of R1 contains data to be output.

```
        LI      R12, PSIBAS     Set up CRU base to point to 9901
        LDCR    @CLKSET, 0      16-bit transfer, set clock to max interval

        LDCR    @INTSET, 7      Enter interrupt mode and enable interrupts 1 – 6

        LI      R12, PSIBAS+32  Set CRU base to I/O ports – output
        LDCR    R1, 8           Output byte from R1, program ports 0 – 7 as output

        LI      R12, PSIBAS+48  Set CRU base to I/O ports – input
        STCR    R2, 8           Store a byte from input port into MSBT of R2

        LI      R12, PSIBAS     Set CRU base to 9901
        SBZ     0               Leave clock mode so decremented contents can be latched
        INCT    R12             Set CRU base to clock read register
        SBO     -1              Enter clock mode
        STCR    R3, 14          Read 14-bit clock read register contents into R3

CLKSET  DATA    >FFFF
INTSET  BYTE    >7E

CLKINT  $                       Clock interrupt service routine - level 3
        LIMI    0               Disable interrupts at CPU
        INC     @COUNT          Count the clock interrupt
        LI      R12, PSIBAS     Set CRU base to point to 9901
        SBZ     0               Enter interrupt mode
        SBO     3               Clear clock interrupt
```

FIGURE . TMS 9900 SAMPLE SOFTWARE TO CONTROL THE TMS 9901

Fig 4.6 TMS 9980 Sample Software to Control the TMS 9901

e) USER I/O PORT

The detailed discription of the user I/O Port is shown in Fig 4.7. The user I/O Port extends from R12 CRU address 000_{16} to $03E_{16}$. The low order four bits (R12 CRU addresses 020 through 026) are also connected to the drivers for light emitting diodes CR1 through CR4 which illuminate in response to a logic one input. The individual bit assignments of the User Port are shown in Table 4.2.

4.1.3 AUDIO CASSETTE INTERFACE (ACI)

An audio cassette recorder can be used as a storage medium for programs used with the TM 990/189. Figure 4.8(a) shows the location of the pins for audio in, audio out and motor control. Figure 4.8(b) indicates the recorder and ACI connections. To dump memory to the cassette, the following instruction is used.

'D <start address> <Sp> <stop address> <Sp>
<entry address> <Sp> IDT= <name> <Sp> READY <Y>'

To load the memory from the cassette, L command is used.

4.1.4 MEMORY ORGANIZATION

With fourteen address lines, the TMS 9980A can address upto 16,384 eight bit memory locations. In this system, a total of 8192 locations are dedicated to onboard devices and the remaining locations are reserved for off-board functions. The off-board memory expansion capability is

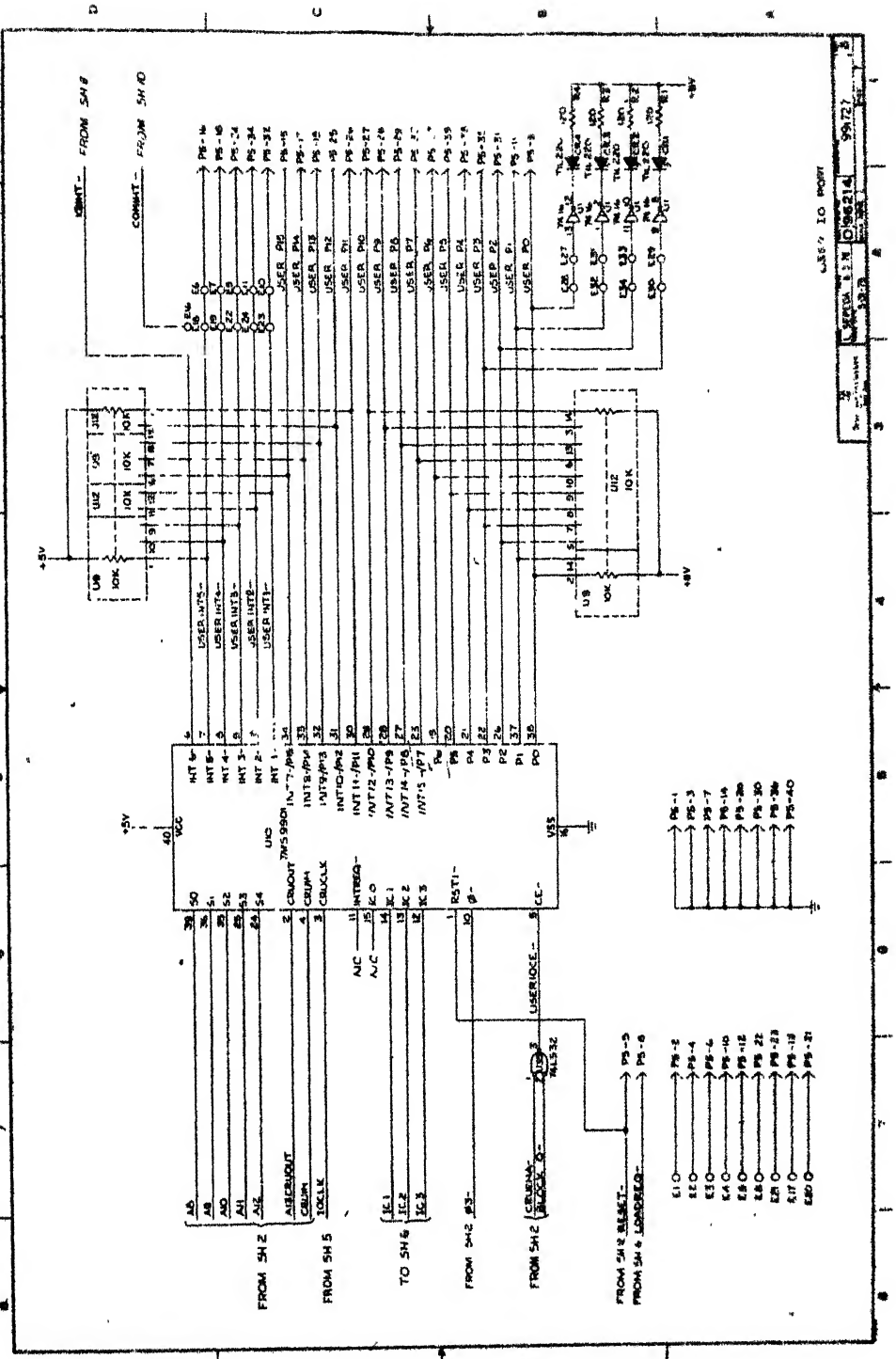


Fig 4.7 User I/O Port

TABLE 8-6. USER PORT I/O MAP

9900 CRU Address	TMS 9901 BIT ASSIGNMENT			SIGNAL LINE AFFECTED
	BIT ADDRESSED	INPUT	OUTPUT	
0400	0	CONTROL BIT	CONTROL BIT	
0402	1	INT1—, CLK 1	MASK 1, CLK 1	UINT 1—
0404	2	INT2—, CLK 2	MASK 2, CLK 2	UINT 2—
0406	3	INT3—, CLK 3	MASK 3, CLK 3	UINT 3—
0408	4	INT4—, CLK 4	MASK 4, CLK 4	UINT 4—
040A	5	INT5—, CLK 5	MASK 5, CLK 5	UINT 5—
040C	6	INT6—, CLK 6	MASK 6, CLK 6	KBINT—
040E	7	INT7—, CLK 7	MASK 7, CLK 7	USER P15
0410	8	INT8—, CLK 8	MASK 8, CLK 8	USER P14
0412	9	INT9—, CLK 9	MASK 9, CLK 9	USER P13
0414	10	INT10—, CLK 10	MASK 10, CLK 10	USER P12
0416	11	INT11—, CLK 11	MASK 11, CLK 11	USER P11
0418	12	INT12—, CLK 12	MASK 12, CLK 12	USER P10
041A	13	INT13—, CLK 13	MASK 13, CLK 13	USER P9
041C	14	INT14—, CLK 14	MASK 14, CLK 14	USER P8
041E	15	INT15—, INTREQ—	MASK 15, RST 2	USER P7
0420	16	P0 INPUT	P0 OUTPUT	USER P0
0422	17	P1 INPUT	P1 OUTPUT	USER P1
0424	18	P2 INPUT	P2 OUTPUT	USER P2
0426	19	P3 INPUT	P3 OUTPUT	USER P3
0428	20	P4 INPUT	P4 OUTPUT	USER P4
042A	21	P5 INPUT	P5 OUTPUT	USER P5
042C	22	P6 INPUT	P6 OUTPUT	USER P6
042E	23	P7 INPUT	P7 OUTPUT	USER P7
0430	24	P8 INPUT	P8 OUTPUT	USER P8
0432	25	P9 INPUT	P9 OUTPUT	USER P9
0434	26	P10 INPUT	P10 OUTPUT	USER P10
0436	27	P11 INPUT	P11 OUTPUT	USER P11
0438	28	P12 INPUT	P12 OUTPUT	USER P12
043A	29	P13 INPUT	P13 OUTPUT	USER P13
043C	30	P14 INPUT	P14 OUTPUT	USER P14
043E	31	P15 INPUT	P15 OUTPUT	USER P15

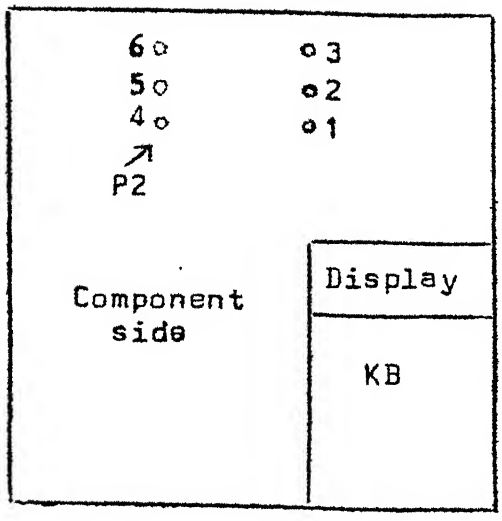
8.6.2 SYSTEM I/O PORT

The System I/O Port occupies CRU addresses 400_{16} through $43E_{16}$ and consists of another TMS 9901 (U11) dedicated to onboard devices such as the keyboard, display, sound disc, and the audio cassette interface. Individual bit assignments for the system port are shown in Table 8-6.

8.6.2.1 Keyboard and Display Interface

Signal flow between the TMS 9901 and keyboard and display is diagrammed in Figure 8-18. UNIBUG software routines scan both the keyboard and display thereby minimizing the hardware required to drive the 80 display segments and read the 45 keyswitches.

The display is a twelve digit common cathode seven segment L.E.D. type of which the middle ten digits are used. Used digits are numbered left to right from 1 to 10. Segments within a digit are designated as shown in Figure 8-19.



P2-1 Motor B GND
P2-2 Audio output GND
P2-3 Audio input GND
P2-4 Motor A
P2-5 Audio output
P2-6 Audio input

Fig 4.8(a): ACI Connector Pins

Recorder	P2 Connections
Auxiliary input	P2-5 (Audio Output) and P2-2(CND)
Earphone Output	P2-6 (Audio Input) and P2-3(GND)

Fig 4.8(b): Recorder And ACI Connections

provided through the Bus Expansion Interface. The system memory map, Fig 4.9, shows the organization of the memory space.

4.2 DATA ACQUISITION AND DISPLAY SYSTEM

The schemes for data acquisition and display systems are discussed in the third chapter. The details of the schemes are explained in the following paragraphs.

4.2.1 SAMPLE AND HOLD

The sample and hold chip (SHM-IC-1) [3] used is shown in Fig 4.10(a). It is a self contained device requiring only an external holding capacitor, the value of which can be chosen according to the requirements of speed and accuracy. The required unity gain, noninverting sample and hold is shown in Fig 4.10(a).

We restrict our input signal magnitude to lie between +5V and -5V. The ADC which follows the S/H needs an analog input of -5V to +5V or 0V to 10V. The 100 PF capacitor gives an acquisition time of 2 micro seconds. To eliminate offset we use a 100 K ohm trimming potentiometer.

4.2.2 ANALOG TO DIGITAL CONVERTER

The analog to digital converter (ADC-EH8B2) [3] used is shown in Fig 4.10(c). The key features of this model are a 2 micro seconds conversion time, unipolar and bipolar operation, parallel and serial outputs and its low cost.

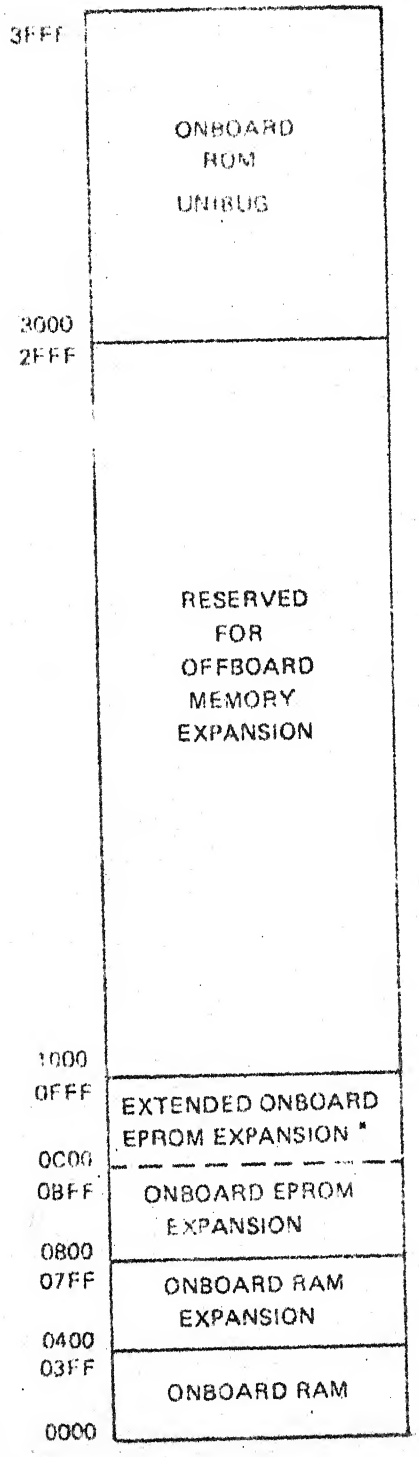


Fig 4.9: System Memory Map

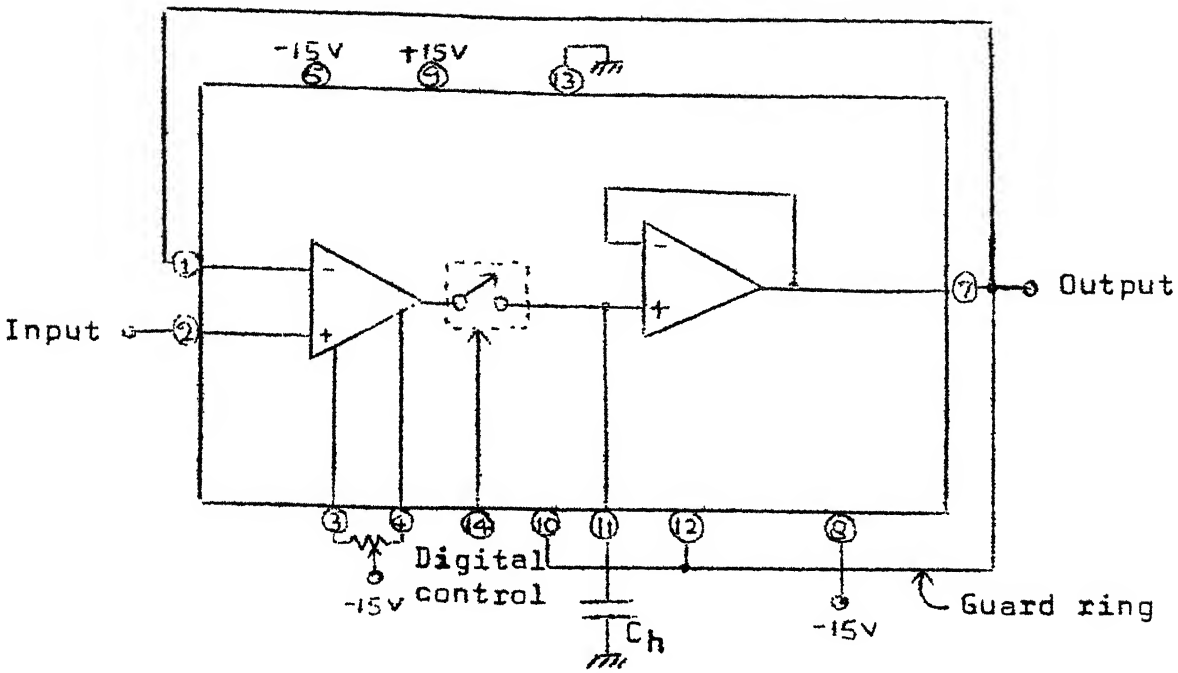


Fig 4.10(a): Model of sample-hold (SHM-IC-1)

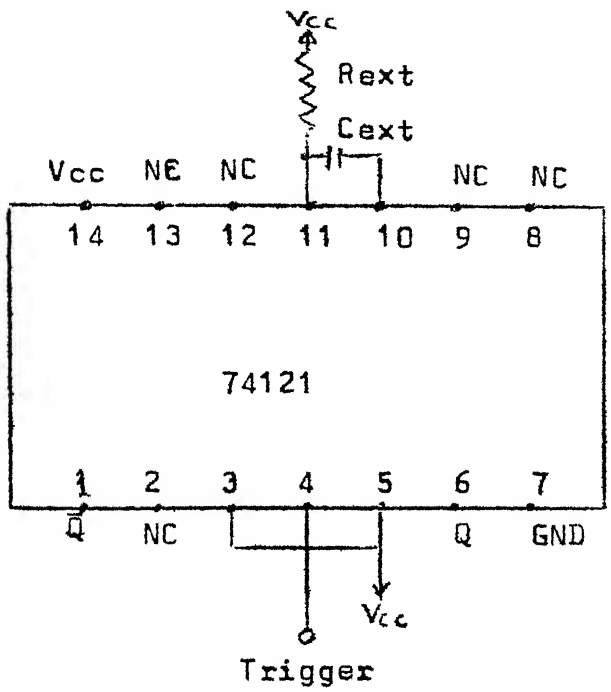


Fig 4.10(b): Pin configuration of monostable

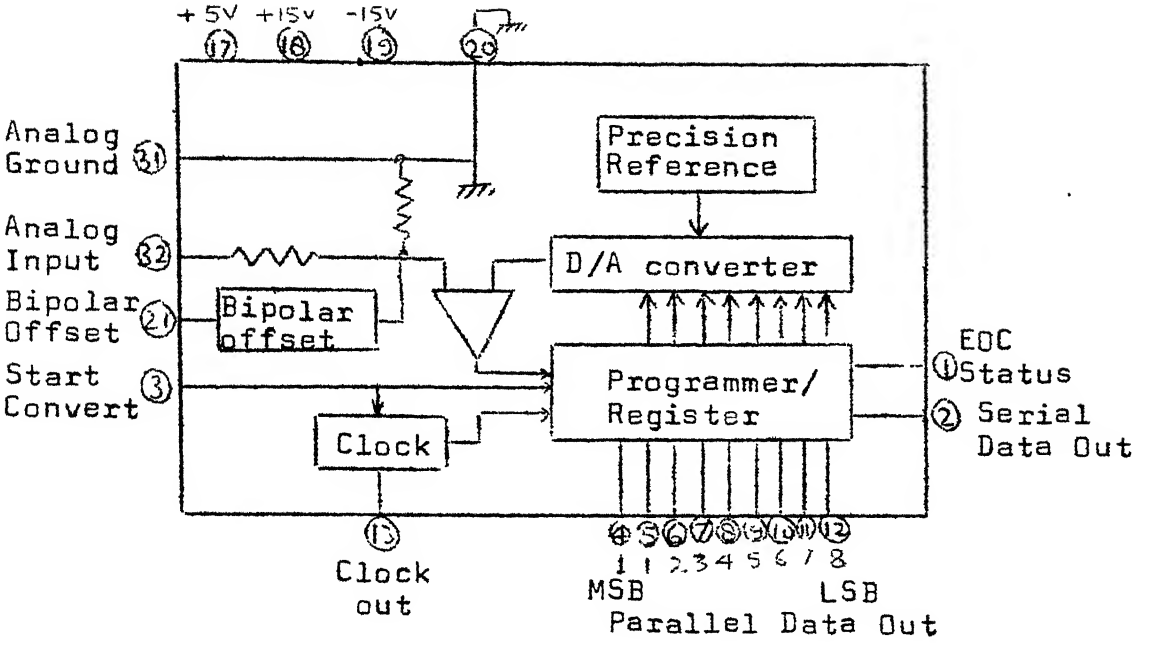


Fig 4.10(c): Model of Analog to Digital Converter (ADC-EH8B2)

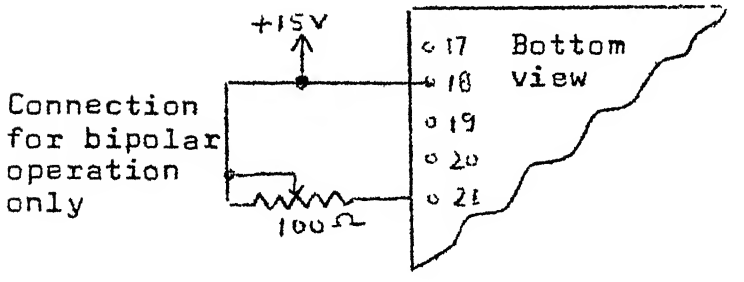


Fig 4.10(d): Bipolar operation of ADC-EH8B2

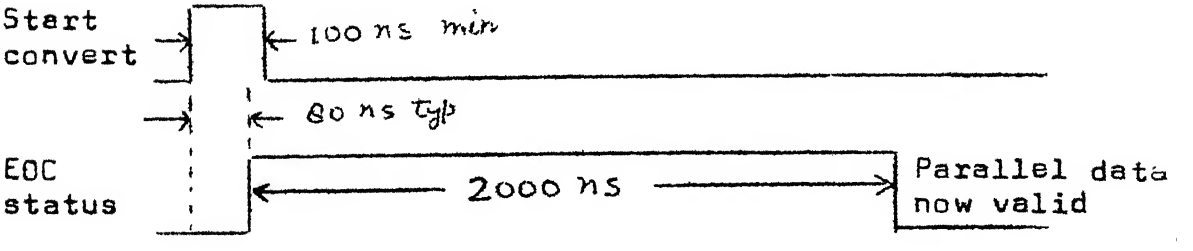


Fig 4.10(e): Timing Diagram for ADC-EH8B2

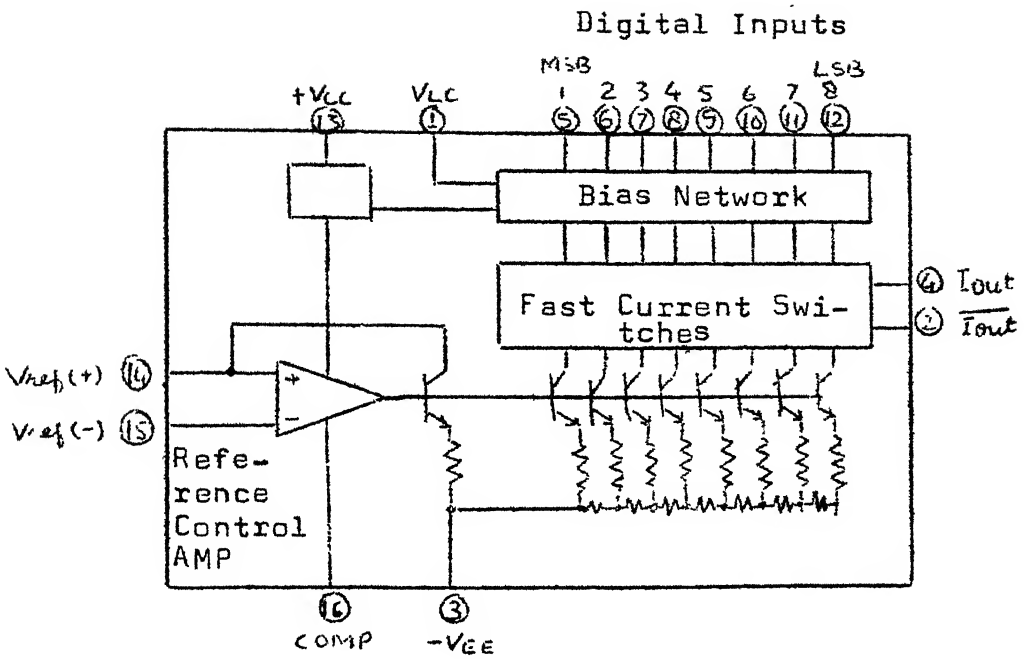


Fig 10(f): Model of Digital to Analog Converter

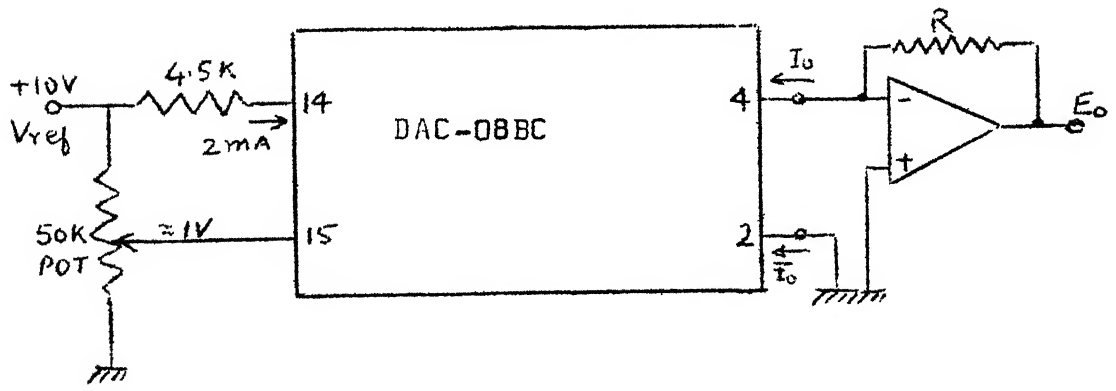


Fig 10(g): Unipolar Operation of DAC

The output can be either in 2's complement or offset binary. Since the input to ADC varies between +5V and -5V. Therefore we configured the convertor to give bipolar operation. For bipolar operation calibration is necessary. We connected the +15V supply to pin 21 of the converter through a 100 ohm trimming potentiometer as shown in Fig 4.10(d). A precision voltage source is connected to pin 32 and the input voltage set to 0.020 volts. The potentiometer is adjusted such that the output code flickers equally between 10000000 and 10000001. The timing diagrams for the analog to digital converter are shown in Fig 4.10(e).

The start convert input signal to ADC is generated by monostable circuit. The configuration of the 74121 is shown in Fig 4.10(b). The output pulse is given by the formula

$$T = Rext$$

The start convert pulse should be at least of 100 nano seconds duration. We choose the pulse width to be 200 nano seconds and

$$Cext = 47\ PF$$

$$Rext = 5.6\ K\ ohm$$

trigger pulse which generates the start convert pulse given by PSI, generated via software of the system.

4.2.3 DIGITAL TO ANALOG CONVERTER

The 8-bit digital to analog converter (DAC-08BC) [3] is shown in Fig 4.10(f). It provides very high speed of 85 nano-seconds coupled with low cost and application flexibility. For unipolar operation, the arrangement is shown in Fig 4.10(g).

CHAPTER 5

SOFTWARE DESIGN

Ever since the development of the first microprocessor in 1971, there has been a tremendous increase in the application of microprocessor in diverse areas such as process control, instrumentation, consumer products, data acquisition and many real time application. The microprocessors have their limitations in scientific and real time application. These limitations mainly arise because of the word length and the speed of available microprocessors.

In this chapter, we will discuss the implementation of FFT on TMS 9980A microprocessor.

5.1 SELECTION OF FIXED-POINT ARITHMETIC

We selected fixed-point arithmetic over floating point arithmetic, because fixed point is faster than floating point through less accurate. In the FFT program utilizing fixed-point arithmetic, the input sequence is scaled such that it can be represented by M bits plus sign and the binary point is assumed to lie to the left of the leftmost magnitude bit. As we move from stage to stage of the program, the magnitudes of the numbers in the sequence generally increase which may cause overflows during different stages of computations. To prevent overflows, scaling is required in fixed-point arithmetic.

5.2 SCALING [4]

The butterfly cycle of the power of two FFT algorithm operates on two complex numbers from the sequence. It takes these two numbers and produces two new complex numbers which replace the original ones in the sequence. Let $X_m(i)$ and $X_m(j)$ be the original numbers. Then, the new pair is given by

$$X_{m+1}(i) = X_m(i) + X_m(j)\ W^P$$

$$X_{m+1}(j) = X_m(i) - X_m(j)\ W^P$$

With the assumption that the binrary point lies at the extreme left, the relationship among the numbers in m^{th} stage and $m+1^{st}$ stage is shown in Fig 5.1. The outside square gives the region of the possible values, $Re[X_m(i)] < 1$ and $I_m[X_m(i)] < 1$. The circle inscribed in this square gives the region $|X_m(i)| < 1$. The circle inscribed in the inside square gives the region $|X_m(i)| < \frac{1}{2}$. Now if $X_m(i)$ and $X_m(j)$ are inside the smaller circle, then $X_{m+1}(i)$ and $X_{m+1}(j)$ will be inside the larger circle and hence not result in an overflow. Consequently, if we control the sequence at the m^{th} stage so that $|X_m(i)| < \frac{1}{2}$, we are certain we will have no overflow at the $m+1^{st}$ stage. However, if $X_m(i)$ and $X_m(j)$ are inside the smaller square, then it is possible for $X_{m+1}(i)$ or $X_{m+1}(j)$ to be outside the larger square and hence result in overflow.

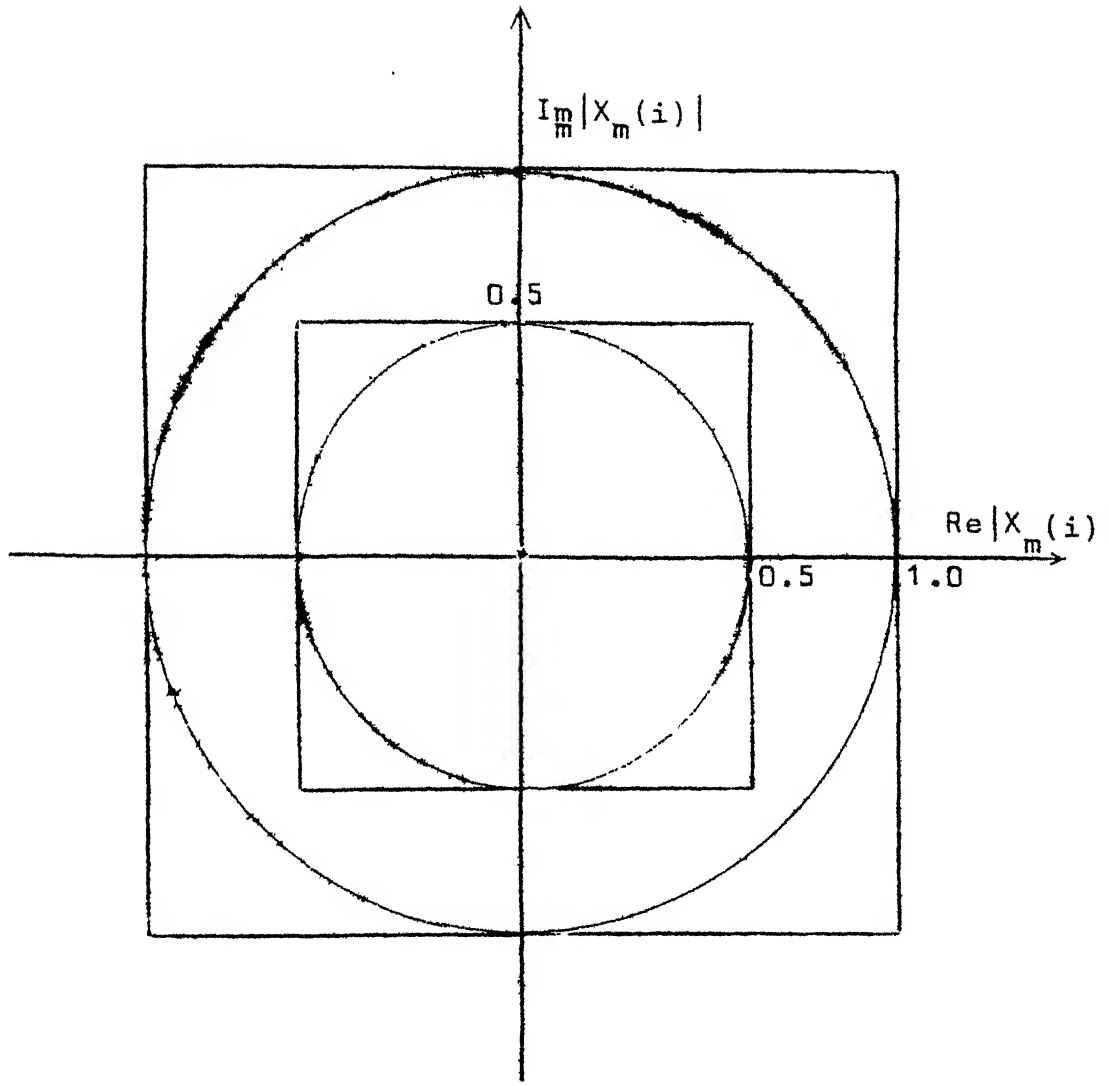


Fig 5.1: Relationship between numbers in m^{th} and m_{+1}^{th} stage.

Three techniques of scaling are given below :

(i) SHIFTING RIGHT ONE BIT AT EVERY ITERATION:

If the initial sequence $X_o(i)$, is scaled so that $|X_o(i)| < \frac{1}{2}$ for all i and if there is a right shift of one bit after every iteration, then there will be no overflow.

(ii) CONTROLLING THE SEQUENCE SO THAT $|X_m(i)| < \frac{1}{2}$

If the initial sequence is scaled so that $|X_o(i)| < \frac{1}{2}$ for all i, then at each iteration we check $|X_m(i)|$ and if it is greater than one half for any i we shift right one bit before calculation throughout the next iteration.

(iii) TESTING FOR AN OVERFLOW

In this case the initial sequence is scaled so that $Re[X_o(i)] < 1$ and $I_m[X_o(i)] < 1$ whenever an overflow occurs in an iteration the entire sequence is shifted right by one bit and the iteration is continued at the point at which the overflow occured.

The first technique is the simplest and easy to adapt for microprocessor. This method gives less accuracy than the other two techniques. In our system, the first technique has been selected.

5.3 ASSEMBLY LANGUAGE CODING OF FFT PROGRAM

The in-place algorithm is slow and requires less memory locations compared to the natural input-output algorithm. Our system has only 2048 bytes of onboard RAM. Therefore the in-place algorithm for 32 sample points has been selected for our system.

The FFT flow chart used for writing a program is shown in Fig 5.2. An assembly listing of the FFT program on the microprocessor TMS 9980A is given in Appendix-A. The details of the program is carried out in the following sections.

5.3.1 INITIALIZATION

To initialize the FFT program, we do following

No. of sample points N = 32

Computation array ℓ = 1

Number of computation array r = 5

Index of the data array K = 0

5.3.2 WEIGHT GENERATION (W)

The values of SINE and COSINE can be generated and stored in a array before entering the FFT. For this purpose only $\frac{N}{4}$ +1 values of SINE from 0 to $\pi/2$ in a step size of $2\pi/N$ are needed and the other values of SINE and all the values of COSINE can be generated from these values. These N/4+1 values of SINE correspond to N/4+1 weights of W, i.e. $W^0, W^1, \ldots . W^{N/4}$.

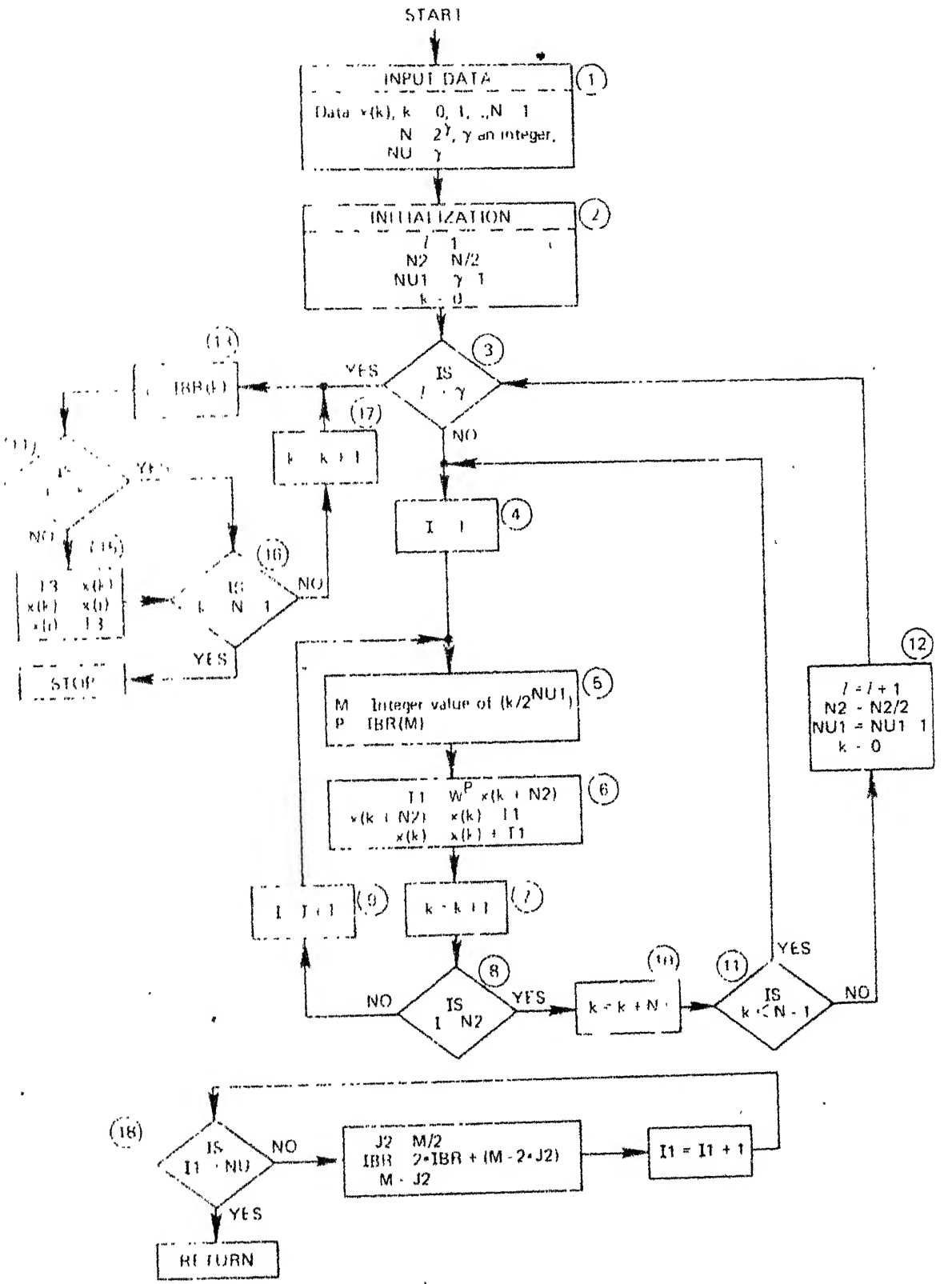


Figure 5-2: FFT computer program flow chart.

For example, for N = 32, 9 values of SINE are stored from 0 to $\pi/2$ in a step size of $2\pi/32$ shown in Fig 5.3. These correspond to W^0, W^1....W^8. Taking the left most bit as sign bit and others as a mantissa, the values of SINE will vary from 0000 0000 to 0111 1111. A set of nine values in hexadecimal is given in Fig 5.3. These values are stored in the system memory (M.A.: 0500-050C) before executing the program.

Location in Memory	Argument of SINE (Radians)	Value of SINE (Hexadecimal)
0	0	0000
1	$(2\pi/32)\times1$	18F8
2	$(2\pi/32)\times2$	30FB
3	$(2\pi/32)\times3$	471C
4	$(2\pi/32)\times4$	5A82
5	$(2\pi/32)\times5$	6A6D
6	$(2\pi/32)\times6$	7641
7	$(2\pi/32)\times7$	7D8A
8	$(2\pi/32)\times8 (= \pi/2)$	7FFF

Fig 5.3: Values of SINE stored in Memory

(a) CALCULATION OF OTHER SINE VALUES (N=32)

For calculating the values of SINE required in the FFT program, following procedure is adopted:

(i) The program should check whether the index is $\leqslant \frac{N}{4}$ or $> \frac{N}{4}$

(ii) If index $\leqslant \frac{N}{4}$, then use the index as it is to fetch the appropriate value of SINE from the array.

(iii) If index $> \frac{N}{4}$, the index is subtracted from $\frac{N}{2}$ and this value is used as the index for fetching the appropriate value of SINE.

(b) CALCULATION OF COSINE VALUES (N=32)

The following procedure is followed

(i) The program checks whether the value of index $\leq \frac{N}{4}$ or $> \frac{N}{4}$

(ii) If index $\leq \frac{N}{4}$, subtract the index from $\frac{N}{4}$ and use this value as index to fetch the value from SINE array

(iii) If index $> \frac{N}{4}$, subtract $\frac{N}{4}$ from the index and use this value as the index to fetch the value from SINE array. This will give appropriate COSINE values.

The software is written to generate all the SINE and COSINE values.

5.3.3 SUBROUTINES

The subroutines which will be called by FFT program are given below :

(a) BIT-REVERSAL ROUTINE (IBR)

The flow chart for bit-reversal operation is shown

by box-18 of the Fig 5.2. The assembly program for the routine is given in Appendix-A.

(b) BUTTERFLY COMPUTATION ROUTINE

The butterfly computation pattern for 8-point FFT is illustrated in Fig 2.1. For an N-point FFT, the basic computation involves the evaluation of the following equations

$$D_1' = D_1 + W^P D_2$$
$$D_2' = D_1 - W^P D_2 \qquad (5.1)$$

where $W = e^{j2\pi/N}$, P is an integer and D_1 and D_2 are complex numbers fetched from sample locations in memory. D_1' and D_2' are the new values generated by the transformation to be stored back in the same locations where D_1 and D_2 were fetched from. The computation D_1' and D_2' from D_1 and D_2 and W^P in accordance with the equation (5.1) is the butterfly computation. This involves the multiplication of two complex numbers and the various steps in the computation are shown below.

Let $D_1 = D_{1R} + D_{1I}$

$D_2 = D_{2R} + D_{2I}$

$W^P = X + jY$

and $W^P . D_2 = A + jB$

then $A = X . D_{2R} - Y . D_{2I}$

$B = Y . D_{2R} + X . D_{2I}$

1) Multiply D_{2R} and X, R1 = D_{2R}.X

2) Multiply D_{2I} and Y, R2 = D_{2I}.Y

3) Add A = R1 - R2

4) Compute $D'_{1R} = D_{1R} + A$

5) Compute $D'_{2R} = D_{1R} - A$

6) Multiply D_{2R} and Y, R3 = D_{2R}.Y

7) Multiply D_{2I} and X, R4 = D_{2I}.X

8) Add B = R3 + R4

9) Compute $D'_{1I} = D_{1I} + B$

10) Compute $D'_{2I} = D_{1I} - B$

11) Store D'_1 and D'_2 in place of D_1 and D_2

(c) OPTIMIZATION OF PROGRAM

For real data inputs, the algorithms discussed in the section 2.8 is more time saving than the generalized algorithm chosen for our system in which the input datas are assumed complex. Since practically our input datas are also real, therefore we tried to optimize the program by exploiting the vary nature of the chosen algorithm. There are r($r=Log_2N$) iterations in the FFT program. In the first iteration, the weight of W is zero, which gives value of X=SIN as zero and Y=COS as one. This means that the components in first array can be computed by simply addition and subtraction and the need for multiplication with SINE and COSINE is eliminated.

For example :

$$D'_{1R} = D_{1R} + D_{2R}$$

$$D'_{1I} = 0 \text{ (Imaginary part is zero)} \qquad (5.2)$$

$$D'_{2R} = D_{1R} - D_{2R}$$

$$D'_{1I} = 0 \text{ (Imaginary part is zero)}$$

The flow chart for computing butterfly cycles is shown in Fig 5.4. The figure indicates that for second iteration, the box-1 is repeated for half of the array as weight of W is zero. For second half of the array, the weight of W equals eight which gives value of X=SIN as one and Y=COS as zero. This means that the remaining components of the array are computed as per the equations given in box-2.

Similarly for third, fourth and etc iterations, the box-3 is computed for any weight of W except the value of 0 and 8. The sequence of computations of box-3 is given in section 5.3.3(b).

The components of the array after computation through box-1, 2 or 3 are divided by 2 before storing back in the system memory.

The square of the magnitude of the imaginary and real parts of each output value after all computation is computed using the following expression

$$Z^2 = X^2 + Y^2$$

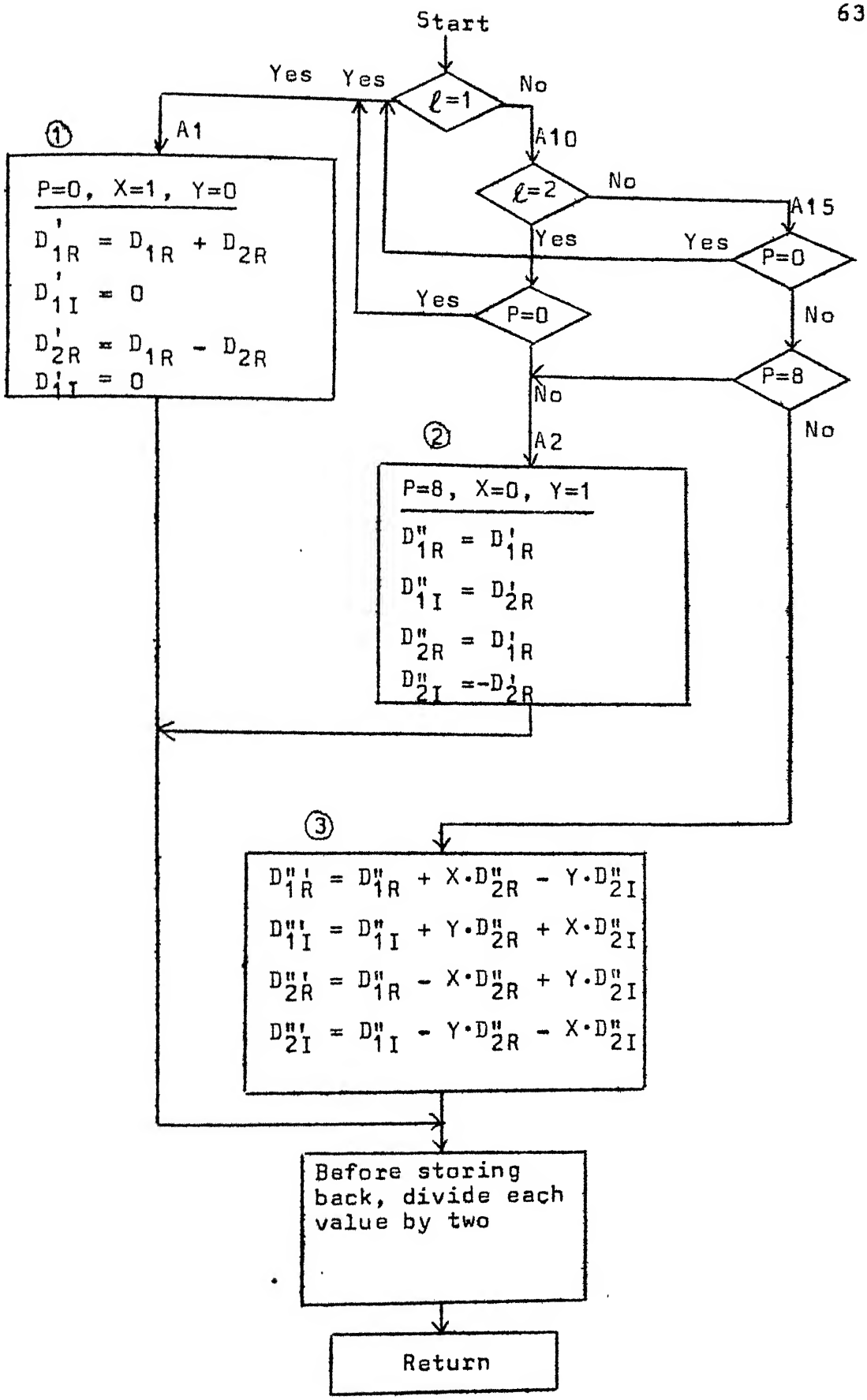


Fig 5.4: Flowchart for computing butterfly cycles

Here we are not interested in the exact value of the output, so the squares of the outputs are computed. The analog form of these values are displayed on the screen through the Z-axis of the oscilloscope.

5.3.4 SOFTWARE FOR THE DATA ACQUISITION AND THE DISPLAY SYSTEMS

The software has been developed and tested on the microcomputer for the data acquisition and the display systems. A set of instructions that have been used for writing this program is shown in Fig 4.6. The used pins of PSI are selected with the help of Table 4.1 and Table 4.2. The listing of the program is given in Appendix-A.

5.4 ROUGH ESTIMATION OF EXECUTION TIME

The rough execution time for one FFT for 32 points can be calculated by observing Fig 5.2 and the program listed in Appendix-A. We assume the average number of clock cycles for each instruction is 20 and the addressing modes are in work-space register modes [5]. The time ($t_c(\emptyset)$) for one clock cycle is 0.400 micro-seconds. The formula used to calculate the execution time (T) for one instruction is

$$T = t_c(\emptyset)\ (C+W\cdot M)\ ;$$

C = Clock cycles

M = 0 (workspace register mode)

The total time taken by butterfly routine	=	50.00 msec.
The total time taken by bit reversal routine	=	145.00 msec.
Time taken by Main program	=	15.00 msec.
Time taken by DAS	=	7.00 msec.
Time taken by Display system	=	7.00 msec.
Total time (approx.)		324.00 msec.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The TMS 9980A microprocessor assembly language program for computing Fast Fourier Transform for a set of 32 points has been developed and tested on TM 990/189 microcomputer.

The data acquisition and the display systems were implemented and tested. A number of clocks for both the systems has been generated via the software of the microprocessor. The frequency of the clock (trigger to the monostable) generated via the software for the data acquisition system is 7.8 KHz. So the sampling frequency is also 7.8 KHz. According to the Nyquist criterion, the sampling frequency should be at least the double of the input signal frequency. Therefore the maximum input signal frequency can be 3.9 KHz.

The eight bits analog to digital converter and eight bits digital to analog converters easily available in the store were used in our project. The eight bits input points are stored in the MSB byte of a word in the memory and after the FFT computation, only the MSB byte of a word is taken out from the system memory and displayed on the oscilloscope.

Hence the accuracy of FFT points is reduced due to round off error. The accuracy can be increased by using 16 bits ADCs and DACs.

The execution time to compute FFT for a set of 32 points is 324 m sec.

Our system is not made for real time computation. The system samples a set of 32 points in a period of 324 m sec, and the information contained in this period is lost. A real time system can be designed if we choose a high speed processor.

6.2 FUTURE WORK

The following related work to spectrum analysis is suggested to improve the performance:

1. The spectrum analyser for real time can be developed.
2. The off-board memory area on TM 990/189 can be used. For this purpose 74LS245 buffers are needed. At present they were not available in the store.
3. The 8086 or any other 16-bits microprocessor can be tried out in place of TMS 9980A.
4. Parallel processor organization instead of sequential scheme may be tried to exploit parallelism in the algorithm.
5. The algorithms discussed in sections 2.4 and 2.8 can be tried out.
6. The spectrum analyser can also be designed with Bit-slice microprocessors.

REFERENCES

1. Brigham, E.O., 'The Fast Fourier Transform'.
2. Rabiner and Gold, 'Theory and Applications of Digital Signal Processing'.
3. Digital Intersil "Engineering Product Handbook" (Manfield: Datel System Inc., 1979)
4. Welch, P.D., "A Fixed-Point Fast Fourier Transform Error Analysis". IEEE Trans. Audio Electroacoustics, 1969.
5. TM 990/189 Microcomputer User's Guide, Texas Instruments, Inc., July 1979.
6. 9900 Family Systems Design and Data Book Microprocessor Series, TI Inc., 1980.
7. FFT: 'Theory, Applications and Hardware' Short term course, Feb 25-March 9, 1976, Dept. of Electrical Engg., IIT Madras.
8. Chatwani, Dilip and Datar, N.N., "Microcomputer Implementation of Real-Time Computation of FFT". B.Tech. Project, April 1982.
9. Stanley, W.D., 'Digital Signal Processing', Reston Publishing Co., Inc., 1975.

APPENDIX-A

ASSEMBLY LISTING

Memory Address	Machine Code	Instruction	Comments
1. Nine values of SINE from 0 to $\pi/2$ in step of $2\pi/32$ stored			
07FE	18F8		
07F2	30FB		
07F4	471C		
07F6	5A82		
07F8	6A6D		
07FA	7641		
07FC	7D8A		
07FE	7FFF		
2. Initial values			
612	0005		value of r
614	0020		value of N
3. Program for data acquisition system			
0510	04CC	CLR R12	
0512	0203	LI R3, > 630	
0514	0630		
0516	04C4	CLR R4	
0518	1D00	SBO 0	clock mode
051A	1E0F	SBZ 15	all I/O becomes in input mode
051C	1E00	SBZ 0	
051E	020C	LI R12, > 0002	
0520	0002		
0522	04C4	CLR R4	
0524	3244	LDCR R4, R9	
0526	1000	NOP	
0528	1D14	SBO 20	P5=1 for ADC enable
052A	020C	LI R12, > 030	
052C	0030		
052E	1EFE	SBZ -2	
0530	020C	yy LI R12, > 030	
0532	0030		
0534	1DFE	SBO -2	clock et P6
0536	1EFE	SBZ -2	
0538	1000	NOP	
053A	04D3	CLR *R3	
053C	3613	STCR *R3, 8	
053E	05C3	INCT R3	R3=R3+2

```
0540    0584        INC R4
0542    0284        CI R4, 32
0544    001F
0546    12F4        JLE >530
0548    1000        NOP
054A    1000        NOP
054C    0204        LI R4, >6F0
054E    06F0
0550    0201        LI R1, >630
0552    0630
0554    0200        LI R8, >8000
0556    8000
0558    C0F1   AA2  MOV *R1+, R3
055A    2008        COC R8, R3
055C    1303        JEQ AA0
055E    CD03        MOV R3, *R4+
0560    04F4        CLR *R4+
0562    1003        JMP AA1
0564    0503   AA0  NEG R3
0566    CD03        MOV R3, *R4+
0568    0734        SETO *R4+
056A    04F4   AA1  CLR *R4+
056C    04F4        CLR *R4+
056E    0284        CI R4, >7F0
0570    07F0
0572    11F2        JLT AA2
0574    1000        NOP
0576    1083        JMP >47E
047E    10D0        JMP >420
0420    10A3        JMP >368
```

4. Main FFT program

```
0368    02E0        LWPI >670
036A    0670
036C    0201        LI R1, 1            > =1
036E    0001
0370    0209        LI R9, >612
0372    0612
0374    C039        MOV *R9+, R0        r=5
0376    04C3        CLR R3
0378    C143        MOV R3, R5          K0
037A    C139        MOV *R9+, R4
037C    C184        MOV R4, R6
037E    0606        DEC R6              R6 = N-1
0380    0914        SRL R4, I           N2 = N/2 = 16
0382    CE44        MOV R4, *R9+        N2, MA=616
0384    0600        DEC R0              r=1=NU1
0386    8801   L1   C R1, @ >612           r
0388    0612
038A    151E        JGT LY
```

```
038C  0202  L2 LI R2, 1              R2=I=1
038E  0001
0390  C0C5  L3 MOV R5, R3            R3=K
0392  0280
0394  0000     CI R0, 0
0396  1301     JEQ A1
0398  0903     SRL R3, 0            M = K / 2^NU1
039A  C803  A1 MOV R3, @ >6B2
039C  06B2
039E  0420     BLWP @ >340
03A0  0340
03A2  0420     BLWP @ >150          Butterfly
03A4  0150
03A6  0585     INC R5               K=K+1
03A8  C285     MOV R5, R10          K→R10
03AA  8102     C R2, R4             I=N2
03AC  1302     JEQ L20
03AE  0582     INC R2               I=I+1
03B0  10EF     JMP L3
03B2  A284  L20 A R4, R10           K=K+N2
03B4  C14A     MOV R10, R5
03B6  018A     C R10, R6            K ≤ N-1
03B8  11E9     JLT L2
03BA  04C5     CLR R5               K=0
03BC  0600     DEC R0               NU1-1
03BE  0914     SRL R4, 1
03C0  C804     MOV R4, @ >616       N2=N2/2
03C2  0616
03C4  0581     INC R1               ℓ=ℓ+1
03C6  10DF     JMP L1
03C3  C805  LY MOV R5, @ >6B2
03CA  06B2
03CC  0420     BLWP @ >340          i=IBR(K)
03CE  0340
03D0  C1E0     MOV @ >61C, R7       i → R7
03D2  061C
03D4  8147     C R5, R7             i ≤ K
03D6  121A     JLE L2
03D8  C285     MOV R5, R10          R5=R10=K
03DA  0A3A     SLA R10, 3           8K
03DC  0209     LI R9, >6F0          Addr. of X(K)
03DE  06F0
03E0  A289     A R9, R10            (6F0+8K) → R10
03E2  C03A     MOV *R10+, R0 )
03E4  C07A     MOV *R10+, R1 )      T3 ← X(K)
03E6  C0FA     MOV *R10+, R3 )
03E8  C11A     MOV *R10,  R4 )
03EA  C207     MOV R7, R8           i → R8
03EC  0A38     SLA R8, 3
03EE  A209     A R9, R8             (8i+6F0) → R8
03F0  064A     DECT R10
03F2  064A     DECT R10
```

```
03F4    064A        DECT R10
03F6    CEB8        MOV *R8+, *R10+
03F8    CEB8        MOV *R8+, *R10+
03FA    CEB8        MOV *R8+, *R10+
03FC    C698        MOV *R8, *R10
03FE    0648        DECT R8
0400    0648        DECT R8
0402    0648        DECT R8
0404    CE00        MOV R0, *R8+        X(i) ← T3
0406    CE01        MOV R1, *R8+        Unscrambled values
0408    CE03        MOV R3, *R8+
040A    C604        MOV R4, "R8
040C    8185    LZ  C  R5, R6
040E    1302        JEQ LP
0410    0585        INC R5
0412    10DA        JMPLY
0414    1000    LP  NOP
```

5. Program for display system.

```
0416    0201        LI R1, >6F0
0418    06F0
041A    0205        LI R5, >630
041C    0630
041E    1001        JMP $+4
0420    10A3        JMP  368
0422    C0B1    A0  MOV *R1+, R2
0424    1000        NOP
0426    C002        MOV R2, R0
0428    3880        MPY R0, R2          X^2
042A    05C1        INCT R1
042C    C0F1        MOV *R1+, R3
042E    1000        NOP
0430    C003        MOV R3, R0
0432    1002        JMP  >438
0434    1031        JMP  >498
0436    1080        JMP  >338
0438    38C0        MPY R0, R3          Y^2
043A    A083        A R3, R2            Z^2=X^2+Y^2
043C    0A22        SLA R2, 2
043E    CD42        MOV R2, *R5+
0440    05C1        INCT R1
0442    0281        CI R1,  >7F0
0444    07F0
0446    11EE        JLT A0
0448    1000        NOP
044A    1000    C0  NOP
044C    0203        LI R3,  >630
044E    0630
0450    04C4        CLR R4
0452    020C        LI P12, >02A
```

```
0454    002A
0456    1E00        SBZ
0458    1000        NOP
045A    020C        LI R12, >030        P5=0 enable display
                                            system
045C    0030
045E    1EFE        SBZ -2
0460    020C    YY  LI R12, >030
0462    0030
0464    1DFE        SBO -2              clock at P6
0466    1EFE        SBZ -2
0468    1000        NOP
046A    3213        LDCR *R3, 8
046C    05C3        INCT R3
046E    0584        INC R4
0470    0284        CI R4, 32
0472    0020
0474    12F5        JLE >460
0476    1000        NOP
0478    1000        NOP
047A    104A        JMP >510
```

6. Sub-routine for Butterfly cycles.

```
0150    0690        LWPI >690
0152    0154
0154    C060        MOV @ >67A, R1      R1=K
0156    067A
0158    C0C1        MOV R1, R3
015A    0A31        SLA R1, 3
015C    1000        JMP $+2
015E    0202        LI R2, >6F0
0160    06F0
0162    A042        A R2, R1            R1=6F0+8K addr. of D1
0164    A0E0        A @ >616, R3        K+N2
0166    0616
0168    0A33        SLA R3, 3
016A    A0C2        A R2, R3            6F0+8(K+N2)= addr.
                                                 of D2
016C    0202        LI R2, > 7F0        addr. of W^P
016E    07F0
0170    C2A0        MOV @ >672, R10     R10= ℓ
0172    0672
0174    028A        CI R10,   1         ℓ =1
0176    0001
0178    162A        JNE A10
017A    C131    A1  MOV *R1+, R4
017C    C191        MOV *R1, R6         Sign of D_1R
017E    0208        LI R8, >8000
0180    8000
0182    2188        COC R8, R6
0184    1601        JNE B1              If sign is +ve
```

```
0186    0504        NEG R4
0188    C173   B1   MOV *R3+, R5        D2R
018A    C093        MOV *R3, R2         Sign of D2R
018C    2088        COC R8, R2
018E    1601        JNE B2
0190    0505        NEG R5
0192    C1C4   B2   MOV R4, R7          D1R
0194    020A        LI R10,  >6B2
0196    06B2
0198    CE87        MOV R7, *R10+
019A    CE85        MOV R5, *R10+
019C    CE85        MOV R5, *R10+
019E    C684        MOV R4, *R10
01A0    0420        BLWP @  >4C6
01A2    04C6                            To addition routine
01A4    C460        MOV @ >6BC,*R1
01A6    06BC
01A8    C4E0        MOV @ >6BA,*R3
01AA    06BA
01AC    C120        MOV @ >6B8, R4
01AE    06B8
01B0    C1E0        MOV @ >6B2,R7
01B2    06B2
01B4    0914        SRL R4, 1
01B6    0917        SRL R7, 1
01B8    0641        DECT R1
01BA    0643        DECT R3
01BC    C444        MOV R4, *R1
01BE    C4C7        MOV R7, *R3
01C0    0380        RTWP

01C2    028A   A10  CI R10, 2
01C4    0002
01C6    161A        JNE A15
01C8    C2A0        MOV @ >61C, R10
01CA    061C
01CC    028A        CI R10, 0
01CE    0000
01D0    13D4        JEQ A1
01D2    C131        MOV *R1+, R4        D'1R
01D4    0914        SRL R4, 1
01D6    05C1
01D8    C173        MOV *R3+, R5        D'2R
01DA    0915        SRL R5, 1
01DC    CC45        MOV R5, *R1+        D"1i
01DE    C453        MOV *R3, *R1
01E0    0551        INV *R1
01E2    C2B3        MOV *R3+, R10
01E4    CCC5        MOV R5, *R3+        D"2i
01E6    C4CA        MOV R10, *R3
01E8    0641        DECT R1
01EA    0641        DECT R1
```

```
01EC    0643        DECT R3
01EE    0643        DECT R3
01F0    C4D1        MOV *R1, *R3        Sign of D"2R
01F2    0641        DECT R1
01F4    0643        DECT R3
01F6    C4C4        MOV R4, *R3         D"2R
01F8    C444        MOV R4, *R1             D"1R
01FA    0380        RTWP

01FC    C2A0    A15 MOV @ >61C, R10
01FE    061C
0200    028A        CI R10, 0
0202    0000
0204    13BA        JEQ A1              P=0
0206    028A        CI R10, 8
0208    0008
020A    13E3        JEQ A2              P=8
020C    1567        JGT A20             P > 8
020E    0209        LI R9, 8
0210    0008
0212    624A        S R10, R9           (N/4-Index)
                                        Addr. for COS value
0214    0609        DEC R9
0216    0A19        SLA R9, 1
0218    A242        A R2, R9            Addr. of X=COS
021A    C119        MOV *R9, R4
021C    3913        MPY *R3, R4         D2R.X
021E    0A14        SLA R4, 1
0220    060A        DEC R10
0222    0A1A        SLA R10, 1
0224    A282        A R2, R10           Addr. of Y=SIN
0226    C15A        MOV *R10, R5
0228    3973        MPY *R3+, R5        D2R.Y
022A    0A15        SLA R5, 1
022C    05C3        INCT R3
022E    C19A        MOV *R10, R6
0230    3993        MPY *R3, R6         D2I.Y
0232    0A16        SLA R6, 1
0234    C1D9        MOV *R9, R7
0236    39F3        MPY *R3+, R7        D2I.X
0238    0A17        SLA R7, 1
023A    0208        LI R8, >8000
023C    8000
023E    C253        MOV *R3, R9         Sign of D2I
0240    2248        COC R8, R9
0242    1602        JNE A30
0244    0506        NEG R6
0246    0507        NEG R7
0248    0643    A30 DECT R3
024A    0643        DECT R3
024C    C253        MOV *R3, R9         Sign of D2R
024E    2248        COC R8, R9
```

```
0250    1602        JNE A31
0252    0504        NEG R4
0254    0505        NEG R5
0256    106F        JMP>336
0336    107E        JMP  > 434
0434    1031        JMP  >498

0498    21C8        COC R8, R7
049A    1602        JNE B1
049C    2148        COC R8, R5
049E    1602        JNE B2
04A0    61C5   B1 S R5, R7
04A2    1003        JMP B3
04A4    0507   B2 NEG R7
04A6    A1C5        A R5, R7
04A8    0507        NEG R7
04AA    2108   B3 COC R8, R4
04AC    1602        JNE B4
04AE    2188        COC R8, R6
04B0    1302        JEQ B6
04B2    A106   B4   A R6, R4
04B4    1003        JMP B8
04B6    0504   B6 NEG R4
04B8    6106        S R6, R4
04BA    0504        NEG R4
04BC    C284   B8 MOV R4, R10
04BE    C147        MOV R7, R5
04C0    10BA        JMP  >436
0436    10C0        JMP  > 338
0338    1092        JMP  > 25E
025E    C1B1   A90 MOV *R1+, R6        D1R
0260    C251        MOV *R1, R9        Sign of D1R
0262    2248        COC R8, R9
0264    1601        JNE A50
0266    0506        NEG R6
0268    0207        LI R7, >6B2
026A    06B2
026C    CDC6        MOV R6, *R7+
026E    CDCA        MOV R10, *R7+
0270    CDC6        MOV R6, *R7+
0272    C5C4        MOV R4, *R7
0274    0420        BLWP @ >4C0        Addition routine
0276    04C6
0278    C1A0        MOV @ >6B2, R6
027A    06B2
027C    C120        MOV @ >6B8, R4
027E    06B8
0280    C460        MOV @ >6BC, *R1    Sign of D'1R
0282    06BC
```

```
0284    0641        DECT R1
0286    0914        SRL R4, 1
0288    CC44        MOV R4, *R1+        D'1R
028A    C4E0        MOV @ >6BA, *R3     Sign of D'2R
028C    06BA
028E    0643        DECT R3
0290    0916        SRL R6, 1
0292    CCC6        MOV R6, *R3+        D'2R
0294    1002        JMP >29A
0296    1000        NOP
0298    1000        NOP
029A    05C1        INCT R1
029C    C1B1        MOV *R1+, R6
029E    C251        MOV *R1, R9
02A0    2248        COC R8, R9
02A2    1601        JNE A60
02A4    0506        NEG R6
02A6    C285    A60 MOV R5, R10
02A8    0207        LI R7, >6B2
02AA    06B2
02AC    CDC6        MOV R6, *R7+
02AE    CDCA        MOV R10, *R7+
02B0    CDC6        MOV R6, *R7+
02B2    C5C5        MOV R5, *R7
02B4    0420        BLWP @ >4C6
02B6    04C6
02B8    C1A0        MOV @ >6B2, R6
02BA    06B2
02BC    C160        MOV @ >6B8, R5
02BE    06B8
02C0    C460        MOV @ >6BC, *R1     Sign of D'1R
02C2    06BC
02C4    0641        DECT R1
02C6    0915        SRL R5, 1
02C8    C445        MOV R5, *R1         D'1I
02CA    05C3        INCT R3
02CC    0916        SRL R6, 1
02CE    CCC6        MOV R6, *R3+        D'2I
02D0    C4E0        MOV @ >6BA, *R3     Sign of D'2I
02D2    06BA
02D4    1000        NOP
02D6    0380        RTWP
02D8    1000        NOP
02DA    1000        NOP

02DC    0209    A20 LI R9, 16
02DE    0010
02E0    624A        S R10, R9           (N/2-Index) Addr,
                                        for SIN values (Y)
```

```
02E2    0609        DEC R9
02E4    0A19        SLA R9, 1
02E6    A242        A R2, R9            Addr. of SIN (Y)
02E8    0200        LI R0, 8
02EA    0008
02EC    6280        S R0, R10           (Index-N/4)
02EE    060A        DEC R10
02F0    0A1A        SLA R10, 1
02F2    A282        A R2, R10           Addr. of COS=X
02F4    1000        NOP
02F6    C11A        MOV *R10, R4
02F8    3913        MPY *R3, R4         D2R.X
02FA    0A14        SLA R4, 1
02FC    C159        MOV *R9, R5
02FE    3973        MPY *R3+, R5        D2R.Y
0300    0A15        SLA R5, 1
0302    C199        MOV *R9, R6
0304    05C3        INCT R3
0306    3993        MPY *R3, R6         D2I.Y
0308    0A16        SLA R6, 1
030A    C1DA        MOV *R10, R7
030C    39F3        MPY *R3, R7         D2I.X
030E    0A17        SLA R7, 1
0310    0208        LI R8,  >8000
0312    8000
0314    0553        INV *R3
0316    C253        MOV *R3, R9
0318    2248        COC R8, R9
031A    1602        JNE A75
031C    0507        NEG R7
031E    1001        JMP A76
0320    0506    A75 NEG R6
0322    0643    A76 DECT R3
0324    0643        DECT R3
0326    0553        INV *R3
0328    C253        MOV *R3, R9
032A    2248        COC R8, R9
032C    1602        JNE A77
032E    0504        NEG R4
0330    1001        JMP A78
0332    0505    A77 NEG R5
0334    1090        JMP A31
```

7. Sub-routine for Addition and Substraction.

```
04C6    06B0        LWPI  >6B0
04C8    04CA
04CA    0208        LI R8,  >8000
04CC    8000
04CE    2048        COC R8, R1
```

```
04D0    1604        JNE B1
04D2    2088        COC R8, R2
04D4    1609        JNE B2
04D6    6042        S R2, R1
04D8    1001        JMP A1
04DA    6042    B1  S R2, R1
04DC    2048    A1  COC R8, R1
04DE    1602        JNE A2
04E0    0501        NEG R1
04E2    0705        SETO R5
04E4    04C5    A2  CLR R5
04E6    1003        JMP B3
04E8    0501    B2  NEG R1
04EA    A042        A R2, R1
04EC    0705        SETO R5
04EE    2108    B3  COC R8, R4
04F0    1606        JNE B4
04F2    20C8        COC R8, R3
04F4    1606        JNE B5
04F6    0504        NEG R4
04F8    6103        S R3, R4
04FA    0706        SETO R6
04FC    1008        JMP B6
04FE    A103    B4  A R3, R4
0500    1001        JMP A3
0502    A103    B5  A R3, R4
0504    2108    A3  COC R8, R4
0506    1602        JNE A4
0508    0504        NEG R4
050A    0706        SETO R6
050C    04C6        CLR R6
050E    0380        RTWP
```

8. Sub-routine for Bit reverse (IBR).

```
0340    06B0        LWPI  >6B0
0342    0344
0344    0206        LI R6, 1            I1=1
0346    0001
0348    04C7        CLR R7              IBR=0
034A    C081    L9  MOV R1, R2          R1=M=J1
034C    0912        SRL R2, 1
034E    C242        MOV R2, R9
0350    0A19        SLA R9, 1           2.J2
0352    0A17        SLA R7, 1           2.IBR
0354    6042        S R9, R1
0356    A1C1        A R1, R7
0358    C042        MOV R2, R1
035A    0586        INC R6
035C    8806        C @>612, R6         I1≤ r
035E    0612
0360    12F4        JLE L9
0362    C807        MOV R7, @ >61C      P=61C
0364    061C
0366    0380        RTWP
```